УДК 519.6

Р. В. Усков

О НЕКОТОРЫХ ОСОБЕННОСТЯХ ПРИМЕНЕНИЯ ТЕХНОЛОГИИ CUDA ДЛЯ МОДЕЛИРОВАНИЯ ПЕРЕНОСА ИЗЛУЧЕНИЯ

Рассмотрены особенности применения графических процессоров общего назначения (на примере технологии nVidia© CUDA) для распараллеливания вычислений. Обсуждаются вопросы программной реализации параллельных алгоритмов, в частности эффективное использование потоковых процессоров и памяти видеоадаптера. Эффективность распараллеливания вычислений показана на примере решения задачи переноса гамма-излучения в веществе.

E-mail: Roman.uskov@gmail.com

Ключевые слова: графические процессорные устройства GPGPU, CUDA, перенос гамма-излучения.

Архитектура современных графических ускорителей обусловлена тем, что вычисления, необходимые для прорисовки картинки, являются однотипными и носят весьма специфический характер. Именно за счет узкой специализации удалось добиться создания устройств с большими вычислительными способностями и относительно небольшой стоимостью. В настоящее время вычислительная мощность современных видеоускорителей в десятки и даже сотни раз превышает аналогичные показатели центральных процессоров при примерно равной стоимости.

Долгое время эти мощности были доступны лишь для разработки графических приложений и не могли использоваться для решения прикладных программ. Однако потенциал в данной области был очевиден, поэтому предпринимались многочисленные попытки приспособить видеоускорители для решения прикладных задач.

В настоящее время существует целая область информатики — GPGPU (General-purpose graphics processing units), — посвященная технике использования графического процессора видеокарты для общих вычислений. Разработан ряд технологий, позволяющих использовать видеоускоритель в прикладных задачах. Одной из них является nVidia©CUDA. В последнее время появились разработки в области моделирования переноса излучения на графических процессорах с использованием данной технологии. Так, для построения реалистичных радиографических образов в рентгеновской медицине создан проект MC-GPU (Monte Carlo simulation of X-ray transport in a GPU with CUDA, 2009) [1], в основе которого лежит методика моделирования

переноса рентгеновского излучения методом Монте-Карло. Для описания объектов используется “воксельное” (объемное) представление, а физическая модель процессов взаимодействия фотонов с веществом берется из пакета PENELOPE 2006. Достигнуто 10–25-кратное ускорение по сравнению с расчетами на центральном процессоре. Данный проект находится в начальной стадии разработки и далек от завершения.

В настоящей работе рассмотрены особенности применения технологии nVidia©CUDA для распараллеливания вычислений с использованием графических процессоров. Обсуждаются особенности программной реализации технологии, в частности подходы к использованию потоковых процессоров и памяти видеоадаптера. Эффективность распараллеливания вычислений показана на примере решения задачи переноса гамма-излучения в веществе.

Обзор технологии. С программной точки зрения технология CUDA представляет собой расширение языка Си, которое можно представить в виде двух взаимозависимых составляющих. Первая составляющая — средства программирования для создания кода, выполняемого в рамках обычной программы на центральном процессоре (далее CPU или хост). Вторая — средства программирования, позволяющие выполнять части (фрагменты) программы (проводить часть вычислений) на видеоускорителе (GPU).

Первая из указанных составляющих (“управляющая часть”) позволяет проводить с помощью CPU такие операции, как выделение памяти на GPU и копирование данных между GPU и хостом, т.е. служит для организации запусков вычислений на GPU. Кроме того, при необходимости, центральный процессор используется, конечно, и в качестве вычислителя.

Вторая составляющая рассматриваемого расширения Си (“реализующая часть”) дает возможность проводить вычисления непосредственно на GPU.

Далее понадобятся следующие определения [2, 3].

Потоком исполнения, или просто потоком (*thread*), называется совокупность логических и арифметических операций (исполняемый код), которые выполняются в определенной последовательности на вычислительном устройстве. При наличии множества потоков будем предполагать, что они выполняются параллельно, т.е. без предписанного порядка во времени.

Ядро — это совокупность потоков, запускаемая на выполнение с хоста и выполняемая на GPU. Потоки в ядре группируются в так называемые *блоки* (*block*); совокупность блоков представляет собой *решетку* (*grid*).

Все потоки имеют идентичный исполняемый код, поэтому, чтобы отличать потоки ядра друг от друга, каждому присваивается идентификационный номер.

Аппаратная структура видеоадаптера. Основными элементами видеоадаптера являются видеопамять (VRAM) и графический процессор (GPU), который состоит из набора мультипроцессоров.

В отличие от обычных процессоров, имеющих одно устройство контроля исполнения, кэш, и несколько арифметико-логических устройств (ALU), графический процессор состоит из набора мультипроцессоров, каждый из которых имеет свое устройство контроля исполнения (Control), область разделяемой (Cache) памяти, области регистров и значительно расширенный по сравнению с обычным процессором набор ALU (рис. 1, [2]).

Такая структура позволяет добиться не только параллельного выполнения алгоритма на разных мультипроцессорах, но и параллельной обработки данных в рамках одного мультипроцессора.

При запуске ядра блоки распределяются между мультипроцессорами (при нехватке мультипроцессоров ставятся в очередь) и выполняются независимо. С каждым мультипроцессором может быть ассоциирован один и более блоков (в зависимости от ресурсов, используемых блоком). Один блок не может выполняться более чем на одном мультипроцессоре.

Соответствие программных средств и аппаратных устройств при проведении расчетов с использованием графических ускорителей следующее:

— поток выполняется одним из вычислителей мультипроцессора (такой элемент еще называют потоковым процессором);

— блок выполняется мультипроцессором;

— решетка — устройством видеокарта.

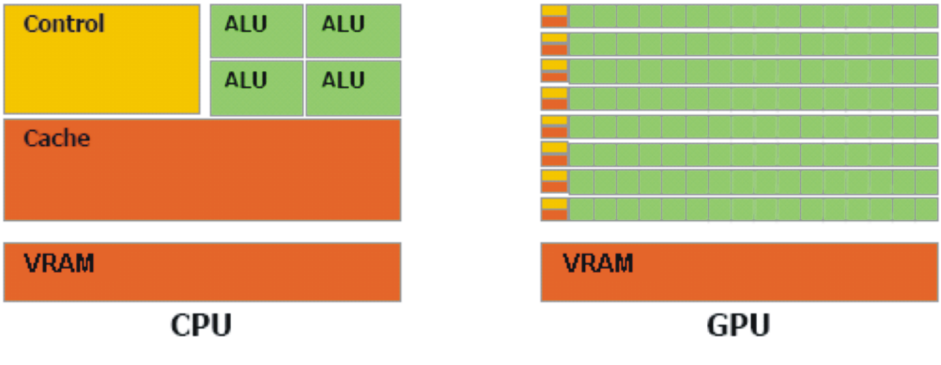

Рис. 1. Иллюстрация отличий структуры обычного процессора (CPU) и графического процессора (GPU)

Хорошо известно [2, 3], что одним из ключевых вопросов при использовании графических процессоров является правильная организация работы с разными типами памяти. Важной особенностью графического адаптера является наличие трех видов памяти: локальной (local), связанной с конкретным потоковым процессором; разделяемой (shared) памяти, доступной в рамках мультипроцессора; глобальной (VRAM) — определенной для видеоадаптера в целом.

Локальная память доступна только потоку, который выполняется на данном вычислителе. Разделяемая память доступна всем потокам блока, а глобальная память доступна всем потокам во всех блоках решетки. Области глобальной памяти могут быть выделены или освобождены только с хоста, т.е. только между вызовами ядер.

Глобальная память, при всем ее удобстве, не кэшируется и является относительно медленной по сравнению с разделяемой памятью. Вообще говоря, использование в процессе решения задачи разделяемой памяти вместо глобальной позволяет добиться значительного увеличения эффективности использования графических процессоров. Особенности использования указанных видов памяти рассмотрены подробно в [2–4].

Особенности разработки приложений для CUDA. CUDA-приложения фактически являются многопоточными приложениями. Однако в силу специфики потоков, а именно большого их числа и крайней легковесности, программирование CUDA-приложений значительно отличается от программирования обычных многопоточных приложений.

Одна особенность состоит в том, что поскольку для задач, под которые изначально создавались современные видеоадаптеры, характерна практически абсолютная декомпозиция данных (т.е. для параллельно обрабатываемых данных результаты обработки одной их части не зависят от результатов обработки другой), в CUDA-приложениях практически отсутствуют методы контроля конкурентного доступа к данным.

Другая особенность также обусловлена архитектурой аппаратных средств, нацеленных на задачи обработки графики, в которых алгоритмы линейны и не содержат условных переходов, т.е. условные переходы не поддерживаются аппаратной частью вовсе. При этом условные переходы и циклы, хотя и поддерживаются программными средствами CUDA, тем не менее являются самыми “тяжелыми” операциями для GPU, поэтому при написании приложений следует старательно их избегать.

Еще одна особенность связана с использованием различных видов памяти. При разработке CUDA-приложений для управления и оптимизации доступны два вида памяти графического адаптера: глобальная

и разделяемая (существует также локальная память, но программист, вообще говоря, не может управлять этим видом памяти). Скорость доступа к глобальной памяти (VRAM) является достаточно низкой, однако эта скорость может быть повышена при помощи связывания нескольких логических запросов к памяти (запросов от разных потоков) в один аппаратный (физический) запрос. Для обеспечения связывания необходимо при организации доступа к памяти следить, по возможности, за выполнением некоторых специальных условий.

При выполнении задания графическим процессором, обработка потоков блока мультипроцессором проводится порциями — *варпами* (warp), при этом варп делится пополам на два полуварпа [2]. Предполагается, что потоки внутри одного полуварпа выполняются одновременно. Размер варпа не является строго определенной величиной и зависит от конкретного оборудования, однако для всех на данный момент существующих устройств размер варпа положен равным 2.

Под связыванием запросов понимается [2] связывание логических запросов к памяти от различных потоков полуварпа в один физический запрос. При этом все запросы должны попадать в одну область памяти определенного размера с начальным адресом, имеющим определенную кратность. Кроме того, номер потока в полуварпе должен соответствовать номеру ячейки в области памяти (рис. 2). На рис. 2 изображено классическое условие обеспечения связанности при работе с четырехбайтовыми величинами. Набор верхних ячеек на этом рисунке соответствует номерам потоков, а нижний набор — номерам ячеек в глобальной памяти видеоадаптера; 16 потоков полуварпа обращаются к 16 идущим подряд ячейкам памяти.

Разделяемая память является более предпочтительной для работы по сравнению с VRAM в основном из-за большей скорости доступа к ней, но при этом ее размер весьма ограничен. Кроме того, необходимо

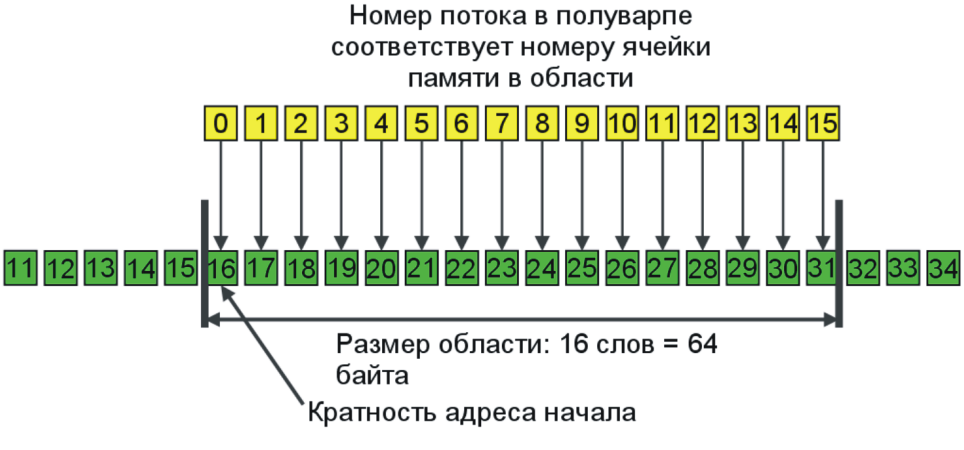

Рис. 2. Условия обеспечения связывания при работе с глобальной памятью

учитывать, что разделяемая память разбита на так называемые банки [2]. При обращении нескольких потоков к ячейкам памяти внутри одного и того же банка происходит так называемый банк-конфликт, который приводит к увеличению времени выполнения запросов к памяти.

Как указывалось выше, правильное использование различных видов памяти крайне важно. Это многократно рассмотрено в документации и примерах, предоставляемых вместе со средой разработки [2], а также в обучающей литературе. Классическим примером является решение задачи о нахождении суммы элементов массива, так называемое reduce, входящее в качестве примера в nVidia© CUDA SDK и подробно рассмотренное в работе [3].

Важность правильного использования памяти можно также показать на практически важном примере, когда требуется решить очень большое число квадратных уравнений. Такие задачи актуальны, например, при моделировании процессов переноса излучения, когда требуется провести трассировку объектов [5], границы которых заданы поверхностями 2-го порядка.

Рассмотрим три варианта решения большого числа квадратных уравнений с использованием графических процессоров:

1) алгоритм не использует разделяемую память. Вся работа ведется в глобальной памяти, причем связанность запросов не обеспечивается;

2) алгоритм использует разделяемую память, однако связывание запросов к глобальной памяти при загрузке из нее данных не обеспечивается;

3) алгоритм использует разделяемую память, и, кроме того, загрузка данных в разделяемую память проводится связанными запросами.

На рис. 3 приведены результаты сравнения эффективности использования графического процессора при выполнении указанных вариантов параллельного кода.

Из рис. 3 следует, что использование разделяемой памяти для хранения массивов коэффициентов уравнений в совокупности с выполнением связанных запросов значительно повышает быстродействие алгоритма.

Гибридные и multi-GPU вычисления. В настоящее время одним из перспективных направлений применения графических ускорителей в прикладных задачах является использование для расчетов нескольких графических процессоров (multi-GPU), а также разработка алгоритмов для проведения параллельных вычислений на гибридных кластерах, содержащих совокупности узлов, каждый из которых состоит из многоядерных CPU и нескольких графических адаптеров.

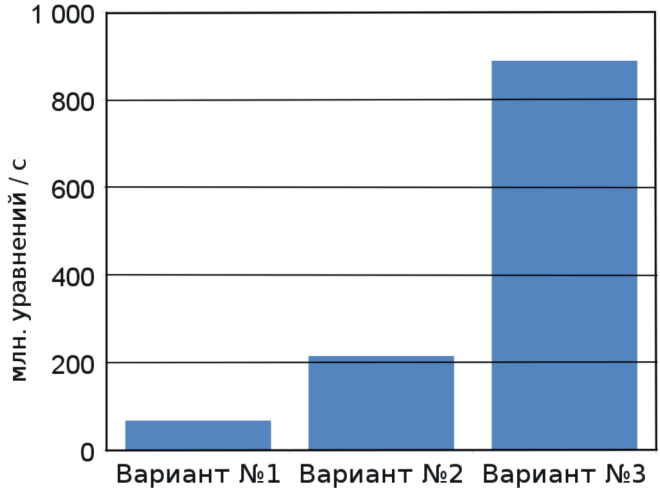

Рис. 3. Сравнение эффективности использования GPU

Причины обращения к построению таких систем хорошо известны — традиционные архитектуры многопроцессорных компьютеров с увеличением числа процессоров получают все больше узких мест. Основным здесь является то обстоятельство, что возможное число транзисторов на один чип значительно превышает сегодня возможности традиционной архитектуры суперкомпьютера.

Одним из таких гибридных устройств, является разработанный в ИПМ им. М.В. Келдыша РАН кластер [6], состоящий из нескольких узлов. Каждый узел представляет собой совокупность восьми процессорных (CPU) ядер и двух видеоадаптеров GeForce GTX 295 с общим числом процессорных (GPU) ядер, равным 960. Узлы объединены сетью на базе интерфейса PCI-E (рис. 4).

Во время подготовки данной статьи в ИПМ им. М.В. Келдыша РАН был запущен в опытную эксплуатацию гибридный суперкомпьютер К-100 с производительностью до 100 Тфлоп/с [8].

Основная идея разработки алгоритмов и программ для таких кластеров заключается в том, чтобы возложить на CPU управление и логическую часть алгоритма, в то время как арифметические вычисления выполнять на GPU и при этом распараллелить как работу CPU, так и расчеты с использованием графических ускорителей. Тем самым можно достичь эффекта “двойного” распараллеливания.

Пример расчета с использованием гибридных кластеров. В качестве примера применения технологии CUDA рассмотрим вычислительный эксперимент, в котором многокомпонентные объекты, имеющие кусочно-гомогенную структуру, облучаются рентгеновским или гамма-излучением.

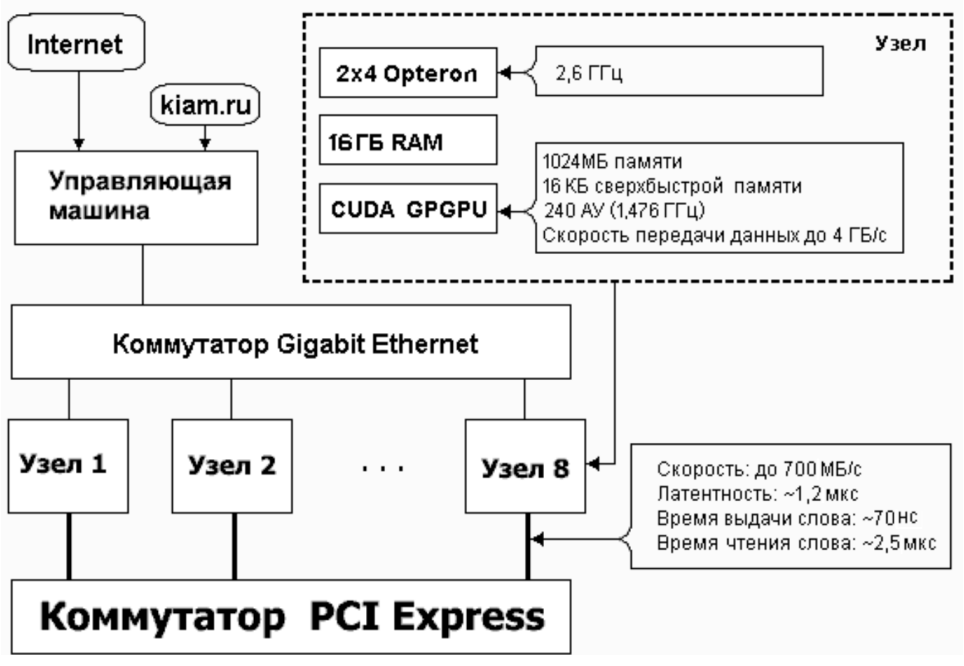

Рис. 4. Структурная схема вычислительного комплекса МВС-Экспресс

Искомой величиной является энергия излучения, поглощенная в фиксированном фазовом объеме $(\Delta\vec{r}, \Delta E)$, играющем роль детектора. Для анализа энергетического распределения поглощенной энергии используется набор таких детекторов.

Процессы взаимодействия гамма-излучения с материалами компонентов объекта описываются на основе общепринятых моделей поглощения и рассеяния квантов излучения в веществе. Для моделирования процессов взаимодействия фотонов излучения с веществом используется модификация метода Монте-Карло. Расчетный алгоритм представляет собой независимое моделирование заданного числа случайных траекторий фотонов с последующей обработкой результатов (оценкой математического ожидания искомого распределения). Каждая траектория состоит из различного числа прямолинейных участков — звеньев — между точками взаимодействия квантов с веществом (рассеяния или поглощения). В процессе моделирования случайной траектории определяется аддитивный вклад каждого звена траектории в общий результат.

Стандартный метод простых испытаний подразумевает розыгрыш одного из вероятных событий (комптоновское или когерентное рассеяние или фотопоглощение) в каждой точке взаимодействия кванта с веществом объекта. Такой подход порождает обилие условных переходов при вычислении характеристик соответствующего процесса и обладает достаточно медленной сходимостью.

Гораздо более эффективным является метод, относящийся к классу весовых модификаций метода Монте-Карло [9], когда полагают, что в каждом узле траектории для кванта реализуются обе возможности — рассеяние и поглощение. При этом поглощенному фотону приписывается вес, равный вероятности поглощения, а рассеянному — вес, равный вероятности "выжить". Такая модификация метода Монте-Карло не только снижает дисперсию результатов, но и уменьшает число условных переходов, что является важным при использовании графических процессоров и связано с особенностями разработки приложений для CUDA. Далее с целью оптимизации работы видеоускорителя в качестве вычислителя была предложена и реализована *многоядерная* схема проведения вычислений. Каждое ядро реализует по возможности однородную совокупность вычислений. Общая схема параллельной реализации рассматриваемого алгоритма изображена на рис. 5.

Анализ алгоритма позволил выделить четыре более или менее однородные части, для каждой из которых строится соответствующее ядро. В левом прямоугольнике на рис. 5 изображено содержание и порядок запуска ядер с центрального процессора на ГПУ.

• *Генерация фотонов* — в рамках данного ядра в каждом из потоков на основе характеристик источника, задаваемых настройками, генерируется текущий фотон. Для него определяются положение, направление движения, начальная энергия и спектральный вес.

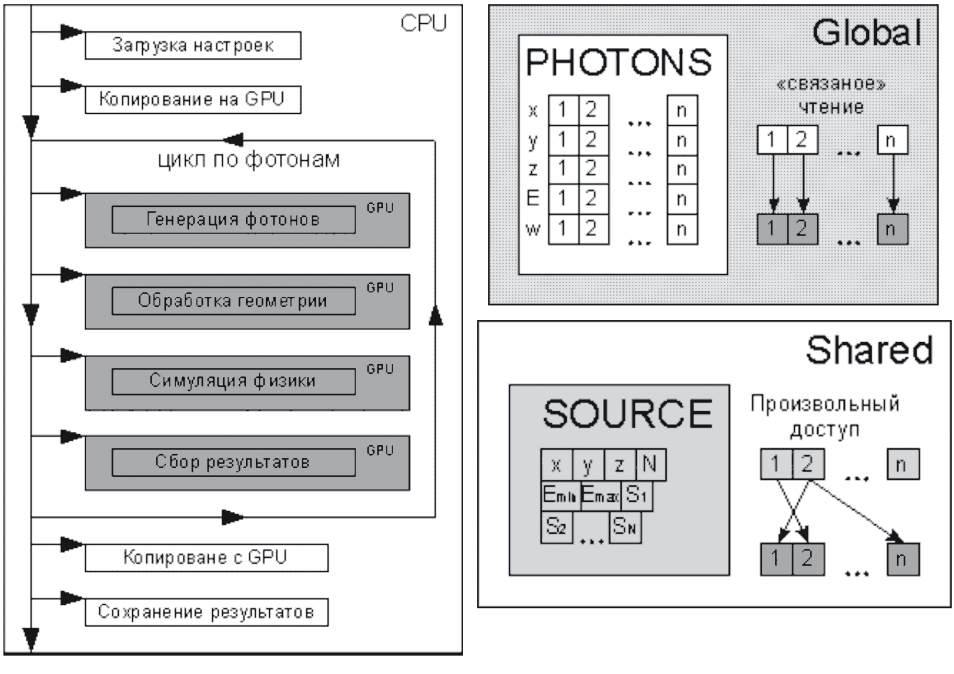

Рис. 5. Схема многоядерной реализации алгоритма

• *Обработка геометрии* — это ядро, в котором все потоки, используя характеристики текущего фотона, а также параметры объекта, задаваемые в настройках, рассчитывают набор интервалов пересечения траектории фотона с однородными компонентами объекта (проводится трассировка объекта).

• *Симуляция физики*. В данном ядре проводится моделирование процессов взаимодействия фотонов с веществом компонентов объекта. В рамках этого моделирования рассчитываются вероятность взаимодействия, условные вероятности различных типов взаимодействия, проводятся розыгрыши необходимых случайных величин. В результате моделирования меняются характеристики текущего фотона, а также сохраняется необходимая информация о поглощенной энергии для каждого канала фотонных энергетических потерь (фотопоглощение, комптоновское рассеяние).

• *Сбор результатов*. На основе энергетических характеристик текущих фотонов, а так же информации о поглощенной энергии, определяется вклад последних взаимодействий фотонов в общий результат. В силу особенностей различных видов памяти видеоадаптера, размещение результатов расчетов в процессе моделирования оказывается нетривиальной задачей. Предложен и реализован эффективный алгоритм сбора промежуточных результатов моделирования, позволяющий, с одной стороны, обеспечить бесконфликтный доступ к памяти и, с другой стороны, достичь максимальной скорости обменных операций.

Предложенная многоядерная схема вычислений позволяет оптимизировать работу с памятью ГПУ (см. ниже), а также минимизировать количество условных переходов в расчетном алгоритме.

В правой части рис. 5 схематично изображена работа с памятью ГПУ. Анализ исходных данных, а также величин, рассчитываемых в процессе моделирования, позволил условно разбить их на две группы — детерминированных данных (т.е. данных, запрос к которым детерминирован и происходит одновременно для всех потоков) и данных со случайным доступом (выбор случайной величины из массива значений и в разное время обращения).

К первой группе данных относятся, например, фотонные характеристики — энергия, координаты, статистический вес, направление движения и другие. Эти данные размещаются в глобальной памяти графического ускорителя. При этом они организуются таким образом, чтобы запросы к ним были связанными.

Данные, к которым предполагался случайный доступ (вероятностные распределения параметров рассеяния и поглощения, характеристики объектов) и для которых невозможно обеспечить связанность

запросов, помещаются в быструю разделяемую память, отлично подходящую для такого рода работы.

Многоядерная структура алгоритма в совокупности с организацией размещения данных позволяет загружать в быструю память одновременно только те данные, которые необходимы на данном этапе алгоритма (в данном ядре), что существенно добавляет свободы при выборе объемов табличных данных, задании числа компонентов объекта и т.п. Это увеличивает потенциальную масштабируемость набора исходной информации.

Оптимизация способа загрузки данных в быструю память GPU была достигнута следующим способом.

Во-первых, размер каждой структуры, загружаемой в эту память, в словах был выравнен путем добавления в конец незначащих байтов до кратного числа потоков в блоке. Во-вторых, структуры разнородных данных представлялись как массивы слов (гомогенизация данных); загрузка осуществлялась порциями, в которых каждый поток отвечал за загрузку единственного элемента, соответствующего его номеру в блоке. После загрузки в быструю память осуществлялся обратный переход к разнородным данным.

Рассмотрим далее результаты моделирования процесса регистрации гамма-излучения в полупроводниковом детекторе. Известно, что регистрирующий энергетический "канал" при использовании таких детекторов определяется энергией, потерянной фотоном в детекторе. Поэтому важным является адекватная оценка спектра поглощенной энергии при взаимодействии излучения с таким детектором.

Если записать связь между энергетическим распределением исходного гамма-излучения $f_\gamma(E)$ и спектром поглощенной энергии $f_{ab}(E)$ в операторном виде [10] как $f_{ab}(E) = Af_\gamma(E)$ и определить функцию $G(E, E')$ с помощью соотношения $G(E, E') = A\delta(E - E')$, то $f_{ab}(E) = \int G(E, E') f_\gamma(E')\, dE'$.

Имея это в виду, легко видеть, что для проведения серии оценок спектральной плотности поглощенной энергии для различных спектров гамма-излучения достаточно один раз рассчитать функцию $G(E, E')$.

Пример такого расчета с использованием описанных выше кластеров представлен на рис. 6, *а*. На этом рисунке показана структура распределения поглощенной энергии по энергетическим ячейкам $\{\Delta E, \Delta E'\}$, которая является графическим изображением функции $G(E, E')$.

С помощью рассчитанной функции $G(E, E')$ путем сверки с исходным спектром фотонов $f(E_\gamma)$ получен спектр поглощения рентгенов-

ского излучения. На рис. 6, *б* изображен график спектральной плотности поглощенной энергии тормозного излучения, которое формируется в рентгеновской трубке с вольфрамовой мишенью при напряжении на трубке 120 кВ.

Результаты расчетов соответствуют предварительным интегральным оценкам поглощенной энергии. Проведенные исследования позволяют оценить, в частности, спектральные характеристики рождающихся внутри материала объекта быстрых фото- и комптоновских электронов.

Время выполнения расчетов функции $G(E, E')$ с погрешностью 1 % составило около 48 ч на CPU с тактовой частотой 3 ГГц. По сравнению с расчетом на одном CPU получено ускорение расчетов: при использовании одного видеоадаптера nVidia GeForce GTX 275 в ∼80 раз, при использовании ускорителя nVidia Tesla в ∼320 раз, при ис-

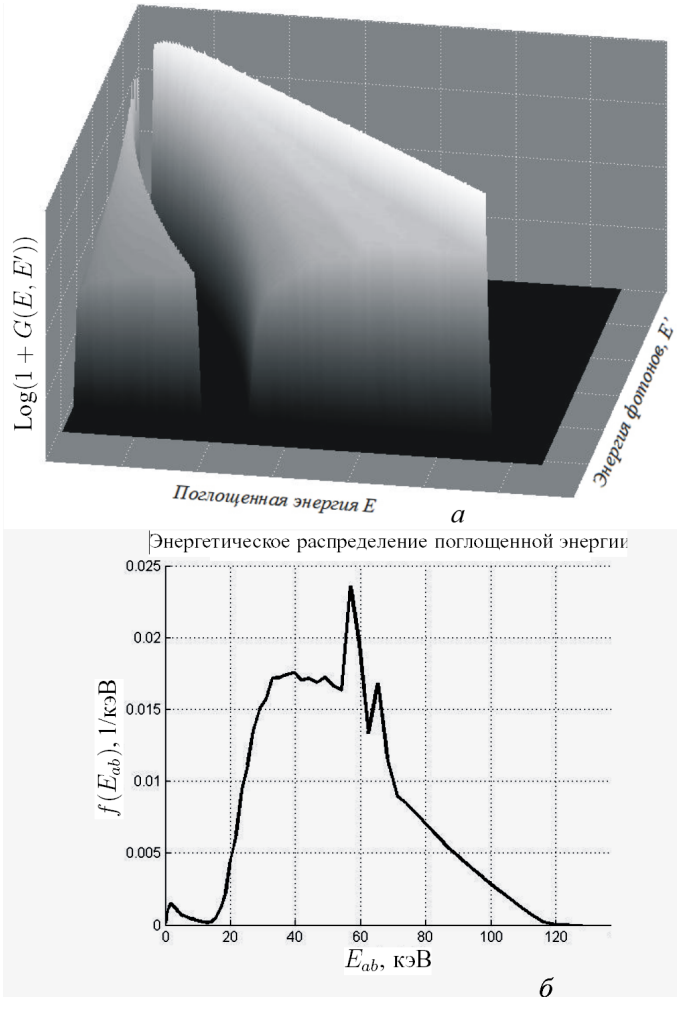

Рис. 6. Графическое изображение функции $G(E, E')$ (*а*) и спектральное распределение поглощенной энергии (*б*)

пользовании четырех узлов гибридного кластера МВС-Экспресс — в ~570 раз.

В заключение отметим, что анализ результатов параллельных вычислений на гибридном кластере МВС-Экспресс показал высокую эффективность рассмотренных подходов к построению алгоритмов моделирования переноса гамма-излучения с применением графических процессоров, а также перспективность использования гибридной вычислительной техники в прикладных исследовательских задачах.

Работа выполнена при поддержке Программы фундаментальных исследований Президиума РАН № 14 "Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация".

СПИСОК ЛИТЕРАТУРЫ

1. http://code.google.com/p/mcgpu/
2. NVIDIA CUDA Programming Guide, Version 2.3.1. 2009.
3. Боресков А. В., Харламов А. А. Основы работы с технологией CUDA. – Москва, 2010.
4. Жуковский М. Е., Усков Р. В. О применении графических процессоров видеоускорителей в прикладных задачах // Препринт Ин-та прикладной математики им. М.В. Келдыша РАН, № 2, 2010.
5. Zagonov V. P., Podolyako S. V., Tillack G. -R., Bellon C. Fast 3D ray tracer algorithm for cone beam radiography // 3-я Междунар. конф. "Компьютерные методы и обратные задачи в неразрушающем контроле и диагностике". Москва 2002.
6. Вычислительная система МВС-Экспресс, http://www.kiam.ru/MVS/research/mvs_express.html.
7. Жуковский М. Е., Усков Р. В. О применении графических процессоров видеоускорителей в прикладных задачах. Ч. II. Моделирование поглощения гамма-излучения // Препринт Ин-та прикладной математики им. М.В. Келдыша РАН. № 20, 2010.
8. http://corp.cnews.ru/news/top/index.shtml?2010/12/31/422015
9. Михайлов Г. А. Весовые алгоритмы статистического моделирования. ИВМиМГ (ВЦ) СО РАН, 2003.
10. Жуковский М. Е., Подоляко С. В., Йениш Г. -Р., Скачков М. В. О моделировании экспериментов с проникающим излучением // Математическое моделирование. – 2007. – Т. 19, № 5. – С. 72–80.

Статья поступила в редакцию 2.11.2010

Роман Владимирович Усков окончил в 2005 г. МГУ им. М.В. Ломоносова. Младший научный сотрудник Института прикладной математики им. М.В. Келдыша РАН. Автор 7 научных работ в области математического моделирования процессов, статистического моделирования процессов взаимодействия ионизирующего излучения с веществом.

R.V. Uskov graduated from the Lomonosov State University in 2005. Junior researcher of the Keldysh Institute for Applied Mathematics, RAS. Author of 7 publications in the field of statistical modeling of the ionizing radiation transport in matter.