«Оптика атмосферы и океана», **30**, № 11 (2017)
DOI: 10.15372/AOO20171103

УДК 551.521.31

Моделирование переноса солнечного излучения в облачной атмосфере методом Монте-Карло с использованием графического процессора и технологии NVIDIA CUDA

Т.В. Русскова*

Институт оптики атмосферы им. В.Е. Зуева СО РАН
634055, г. Томск, пл. Академика Зуева, 1

Поступила в редакцию 13.06.2017 г.

Обсуждаются вопросы о повышении эффективности численного моделирования распространения солнечного излучения в атмосфере Земли методом Монте-Карло путем перехода от последовательных вычислений к параллельным. Представлен новый параллельный алгоритм метода, ориентированный на вычислительную систему с графическим процессором, поддерживающим технологию NVIDIA CUDA. Эффективность распараллеливания проанализирована на примере расчета потоков нисходящей и восходящей солнечной радиации как в вертикально однородной, так и неоднородной моделях атмосферы. Представлены результаты апробации алгоритма в различных атмосферных условиях, в том числе в условиях однослойной и многослойной сплошной облачности, с учетом и без учета селективного молекулярного поглощения. Анализируются результаты тестирования кода на видеокартах с разной вычислительной мощностью. Показано, что перенос вычислений с традиционных ПК на архитектуру графических процессоров дает более чем стократный прирост производительности и в полной мере раскрывает возможности используемой технологии.

Ключевые слова: метод Монте-Карло, потоки солнечной радиации, параллельные вычисления, GPU, технология CUDA, ускорение вычислений; Monte Carlo method, solar radiation fluxes, parallel computing, GPU, CUDA technology, computation speedup.

Введение

Бурное развитие технологий дистанционного зондирования Земли и расширение технических возможностей информационно-измерительных систем способствуют интенсификации использования получаемых данных. С помощью постоянно совершенствуемой аппаратуры, размещенной как на Земле, так и на спутниковых носителях, происходит непрерывное накопление больших объемов информации, требующей быстрой обработки для ее корректной и своевременной интерпретации при решении задач мониторинга окружающей среды, прогноза погоды и климата. Современные методы восстановления различных характеристик атмосферы (аэрозоль, газ, облачность) и подстилающей поверхности (отражательные характеристики) в оптическом спектральном диапазоне, как правило, предполагают решение уравнения переноса солнечного излучения (УПИ) в соответствующих эксперименту условиях (см., например, [1]).

За последние десятилетия для решения УПИ разработано большое количество численных, аналитических и приближенных методов [2]. «Золотым стандартом» численного моделирования распространения излучения в среде является метод Монте-Карло [3, 4], полное теоретическое обоснование которого представлено в работах Г.И. Марчука и др. [3]. Статистический подход, лежащий в основе метода, обусловливает его гибкость (метод практически не накладывает ограничений на геометрию рассматриваемых систем) и сравнительную простоту реализации: моделирование переноса каждого отдельного фотона осуществляется практически так же, как он распространяется в среде – рассеивается, поглощается, отражается и преломляется – с использованием простых физических законов. Однако моделирование с приемлемой точностью многих миллионов таких фотонов для того или иного диапазона изменения входных параметров может привести к многочасовым расчетам, что при всех достоинствах метода ограничивает его использование в задачах, требующих оперативного решения. Еще более высокая ресурсозатратность метода проявляется при решении УПИ в сложных условиях численного эксперимента (разорванная либо оптически плотная сплошная облачность, полосы сильного молекулярного поглощения, сумерки и т.п.), когда время моделирования стремительно возрастает, что не всегда приемлемо для решения задач даже на исследовательском этапе.

Стохастичность метода Монте-Карло обусловливает необходимость генерации псевдослучайных чисел практически на каждом шаге статистического

* Татьяна Владимировна Русскова (btv@iao.ru).

алгоритма (моделирование длины свободного пробега, расчет нового направления в результате рассеяния или отражения от подстилающей поверхности). Длина генерируемой последовательности псевдослучайных чисел увеличивается с нарастанием сложности численного эксперимента. Например, при расчетах средней интенсивности рассеянной радиации в разорванной облачности к этим операциям добавляется также процедура моделирования случайного числа облаков со случайными горизонтальными и вертикальными размерами. Другим примером являются расчеты в сумеречных условиях, когда для достижения приемлемой точности вычислений необходимо моделировать существенно большее количество траекторий, а значит, и гораздо более длинную последовательность псевдослучайных чисел. Таким образом, существуют условия, при которых генерация псевдослучайных чисел составляет весомую долю в общем объеме вычислений. Это вынуждает предъявлять особые требования к качеству генераторов псевдослучайных чисел (ГПСЧ), в частности к их быстродействию и периоду.

Благодаря прогрессу вычислительной техники и процессоров в частности, которые на протяжении последних десятилетий эволюционировали по закону Мура [5], скорость вычислений с момента появления метода значительно возросла. Однако время исполнения последовательной программной реализации метода на одном CPU (Central Processing Unit) стремится к некоторому пределу, переход на CPU с улучшенными характеристиками уже не дает существенного прироста производительности.

Смена парадигмы вычислительного процесса, обязанная появлению технологий параллельного программирования, привела к созданию подходов, благодаря которым моделирование переноса излучения методом Монте-Карло стало возможным на нескольких однопроцессорных персональных компьютерах (ПК) [6], на одном ПК с многоядерным процессором [7], на CPU-кластере с использованием технологии MPI [0]. Это позволило сократить время расчетов почти пропорционально количеству одновременно функционирующих вычислительных единиц. В настоящее время массивно-параллельные вычисления в области теории переноса проводятся на дорогостоящих многопроцессорных системах и кластерах (см., например, [9]), а решение отдельных исследовательских задач, как правило, не выходит за рамки использования традиционных ПК, сопровождаясь значительными потерями времени, обусловленными ожиданием завершения расчетов.

Новый этап развития вычислительных алгоритмов связан со стремительной эволюцией графических процессоров GPU (Graphics Processing Unit). Современные GPU обладают несколькими мультипроцессорами, каждый их которых состоит из большого числа потоковых процессоров, способных выполнять примитивные математические операции одновременно. По сравнению с традиционными высокопроизводительными архитектурами вычислительных систем при решении определенного класса задач GPU и кластеры GPU обладают лучшими показателями энергоэффективности и производительности с учетом их стоимости, а также меньшими требованиями к инженерной инфраструктуре [10]. Архитектуру GPU можно кратко охарактеризовать как «макроархитектуру вычислительного кластера, реализованного в микромасштабе» [11].

К настоящему моменту наиболее впечатляющие результаты параллельной программной реализации метода Монте-Карло для численного моделирования распространения излучения в оптически плотной среде на графических процессорах достигнуты в области биомедицинских технологий [12]: интенсивные исследования ведутся как в России [13–15], так и за рубежом [16–21]. Эффективное программное решение, позволившее ускорить расчеты на два порядка, предложено в работах [16, 17]. При моделировании распространения излучения в многослойной трехмерной среде в соответствии с подходом [20] достигнуто трехсоткратное ускорение.

В атмосферной оптике число таких примеров весьма ограничено (если не брать во внимание работы, посвященные решению УПИ с использованием GPU другими методами, например методом дискретных ординат [1, 22]). Лишь в [23] показано, что при расчетах яркости поляризованного солнечного излучения в пределах заданного авторами диапазона изменения параметров достигнуто трехсоткратное ускорение, а учет поглощения излучения атмосферными газами с использованием метода *line-by-line* [24] приводит к снижению эффективности распараллеливания и обеспечивает пятидесятикратное ускорение. Иные результаты реализации метода на графических процессорах для радиационных расчетов в атмосфере Земли среди опубликованных работ автору не известны.

Цель данной работы — тестирование нового параллельного алгоритма метода Монте-Карло, разработанного для оперативного решения задач численного моделирования распространения солнечного излучения в оптически плотной среде на графических процессорах с помощью интерфейса прикладного программирования CUDA (Compute Unified Device Architecture) [25]. Предложенный GPU-код является частью нового программного комплекса GPU-MATHART (**GPU**-accelerated **M**onte C**A**rlo **TH**ree-Dimension**A**l **R**adiative **T**ransfer Codes), представляющего собой вычислительное ядро создаваемой высокопроизводительной информационно-вычислительной системы HiPeCS (**Hi**gh **Pe**rformance **C**omputer **S**ystem).

1. Особенности программной реализации

В данном разделе представлен новый алгоритм метода Монте-Карло для численного моделирования переноса солнечного излучения, программная реализация которого разработана автором на базе [26] для исполнения на вычислительной системе с графическим процессором NVIDIA, поддерживающим архитектуру CUDA.

Написанное на языке Си консольное приложение GPU-MATHART_FLUX позволяет вычислять потоки нисходящего и восходящего солнечного излучения на уровнях подстилающей поверхности и верхней границы атмосферы, а также на промежуточных уровнях между ними как в безоблачной, так и облачной атмосфере. Моделирование переноса излучения осуществляется в плоско-параллельной горизонтально-однородной аэрозольно-молекулярной модели атмосферы с использованием весовой модификации метода Монте-Карло [3]. Отражение излучения от подстилающей поверхности (ПП) моделируется по закону Ламберта. Облачность представлена в виде сплошного плоского слоя. Предусмотрена возможность проведения расчетов в многослойной облачности с разным типом облаков на соответствующих высотах. Свойства аэрозоля и облаков в каждом из слоев описываются стандартными оптическими характеристиками: коэффициентом ослабления, индикатрисой и альбедо однократного рассеяния. Молекулярное рассеяние излучения моделируется как рэлеевское. Учет селективного молекулярного поглощения солнечного излучения осуществляется через функцию пропускания [3], аппроксимируемую конечным рядом экспонент.

В классическом однопоточном исполнении алгоритм метода Монте-Карло представляет собой последовательное моделирование заданного количества независимых случайных траекторий фотонов с последующей обработкой результатов — оценкой математического ожидания искомого распределения и нормализованного относительного среднеквадратического отклонения [27], расчет которого возможен благодаря объединению траекторий в так называемые «пачки». Каждая траектория состоит из различного числа прямолинейных участков между точками взаимодействия фотона со средой. В процессе моделирования случайной траектории определяется аддитивный вклад каждого ее «звена» в общий результат. Моделирование траектории фотона завершается вылетом фотона за пределы среды либо достижением весовым коэффициентом некоторого критического значения. Вычислительная схема моделирования на каждом звене траектории одинакова, результат моделирования определяется состоянием фотона в начале звена.

Параллельная реализация алгоритма с использованием технологии CUDA приводит к довольно существенной переработке исходного кода и модификации самого алгоритма. В вычислительной модели CUDA графический процессор можно рассматривать как сопроцессор к CPU с собственной памятью, на который передаются вычисления для параллельной обработки большого числа потоков. Концепция технологии определяет особенности GPU-реализации алгоритма. Упрощенная блок-схема алгоритма, ориентированного на вычислительные системы с графическим ускорителем, представлена на рис. 1, где N_{ph} — заданное количество фотонов, n_{ph} — текущее количество запущенных фотонов, NP_{ph} — общее количество фотонов в пачке, NS_{GPU} — общее количество звеньев для одного запуска ядра, i_{ph} — счетчик количества звеньев, которые преодолел фотон, j_{p} — счетчик количества обработанных пачек траекторий, w_{ph} — текущий статистический вес фотона, W — предельное значение весового коэффициента.

Алгоритм удовлетворяет основным требованиям, выполнение которых необходимо при разработке высокопроизводительного кода на GPU: преобладание вычислений по отношению к загрузке данных из памяти, малая доля условных операторов, равномерность загрузки вычислительных потоков, локальность данных, достаточное количество вычислительных потоков для полной загрузки планировщиков всех имеющихся мультипроцессоров.

Как и большинство приложений, реализующих возможности графических процессоров, приложение GPU-MATHART_FLUX состоит из двух основных частей (рис. 1). Одна часть исполняется на центральном процессоре и обеспечивает взаимодействие приложения с периферийными устройствами и пользователем (запуск приложения, чтение данных и запись результатов в файл и т.д.), размещение исходных данных и результатов в оперативной памяти компьютера, проведение вспомогательных вычислений, копирование данных в видеопамять и получение из нее результатов расчетов, запуск вычислительного ядра на графическом процессоре.

Другая часть программы — вычислительное ядро — исполняется на графическом процессоре. Ядро осуществляет такие операции, как запись данных в видеопамять, обмен данными с разделяемой памятью, собственно моделирование случайных траекторий с использованием регистров графического процессора, арифметических и логических операций, встроенных в CUDA математических функций, вызовов пользовательских функций с квалификатором *_device*, например таких как *turn*() (моделирование нового направления) и *geom*() (расчет оптического пути фотона), операторов условного перехода и циклов.

Исходные, а также используемые в процессе моделирования данные организованы в структуры *layer_struct*, *photon_struct*, *medium_struct* и *thread_struct*, созданные для хранения информации о каждом слое, текущем состоянии фотона, свойствах среды и промежуточных результатах соответственно.

Важным условием эффективной реализации программы на GPU является правильное использование имеющихся типов памяти. Для решения задачи задействованы регистры, константная и глобальная память.

Данные, не изменяемые в процессе исполнения ядра, могут храниться в специальной и сравнительно небольшой области памяти GPU — константной. Чтение из константной памяти может уменьшить количество операций обмена данными с процессором по сравнению с чтением из глобальной памяти [28]. Поскольку этот тип памяти подходит для хранения данных, общих для всех вычислительных потоков, в приложении GPU-MATHART_FLUX в константную память видеокарты копируется часть элементов введенных структур, в том числе структура *layer_struct*, определяющих в совокупности свойства среды.

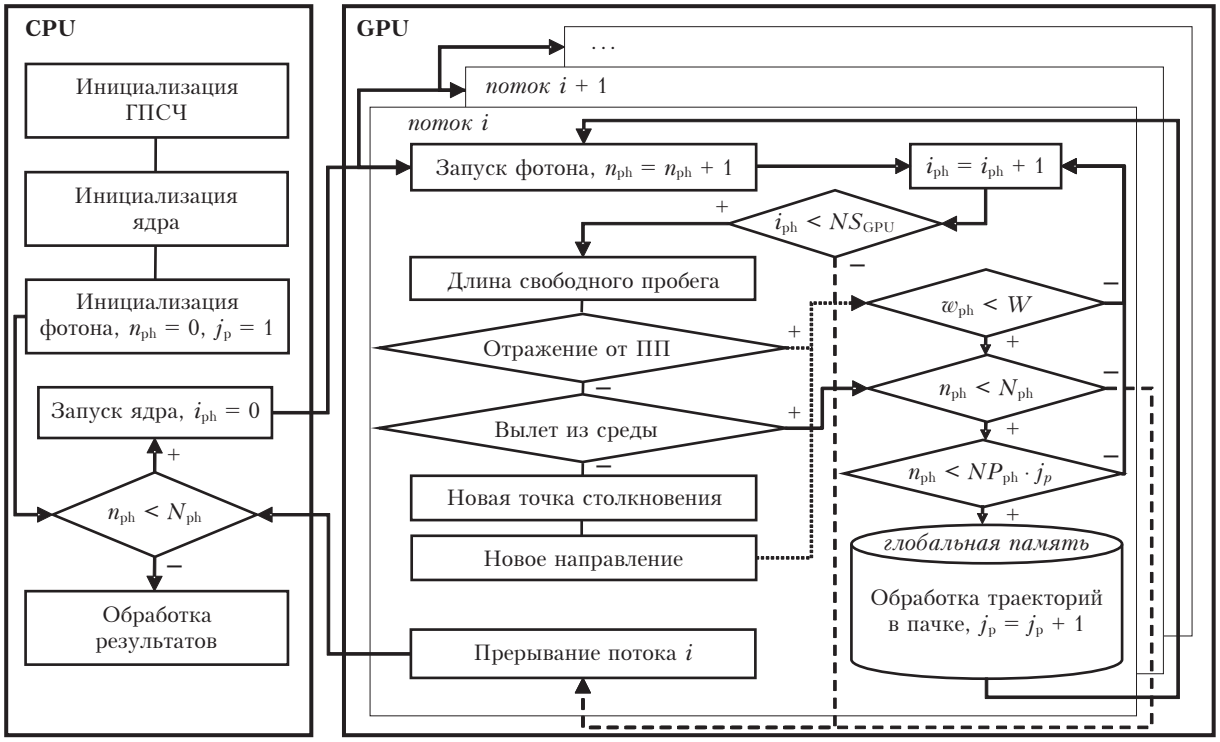


Рис. 1. Общая схема параллельного алгоритма метода Монте-Карло, ориентированного на вычислительные системы с графическим ускорителем

В CUDA нет явных способов использования регистровой памяти — самой быстрой среди других типов: всю работу по размещению данных в регистрах берет на себя компилятор. В регистрах следует располагать данные, используемые наиболее часто. В приложении GPU-MATHART_FLUX таковыми являются некоторые параметры ГПСЧ, а также характеристики моделируемой траектории (текущие координаты и направление движения частицы, статистический вес, текущий номер слоя), принадлежащие структуре *photon_struct*. Время жизни регистров ограничивается временем работы вычислительного ядра.

Разделяемая память и регистры являются быстрыми типами памяти, но имеют небольшой объем, в связи с чем использовать их для работы с результирующими данными не представляется возможным. В GPU-MATHART_FLUX для хранения структуры *thread_struct* используется глобальная память, позволяющая размещать большие объемы данных.

Для выполнения арифметических расчетов на GPU задействованы библиотеки ускоренных функций для CUDA (деления (*_fdividef*(*float*, *float*)), расчета значений логарифма (*_logf*(*float*)), извлечения корня (*_fsqrt*(*float*))). Отметим, что приложение ориентировано на проведение вычислений с одинарной точностью, что тоже существенно ускоряет его работу.

При GPU-реализации алгоритма возникло несколько трудноразрешимых проблем, решение которых в однопоточном приложении тривиально. Одна из них связана с простыми арифметическими операциями над переменными, размещенными в глобальной памяти: при переходе от CPU- к GPU-реализации мы сталкиваемся с непредсказуемостью результата, когда несколько потоков одновременно читают или изменяют разделяемые значения. Поскольку выполнение операции не может быть разбито на части во избежание вмешательства другого потока, возникает необходимость использования атомарных операций. В программе GPU-MATHART_FLUX это такие операции, как *atomicAdd*() и *atomicExch*(). Атомарные инструкции гарантируют согласованность данных, позволяя только одному потоку обновлять значение разделяемой переменной в любой момент времени; при этом блокируются другие потоки, которым требуется доступ к той же переменной, что обусловливает более высокие временные затраты по сравнению с традиционными операциями с памятью. Контроль за актуальностью данных в общедоступной разным потокам для чтения и записи области памяти осуществляется с помощью функции *_syncthreads*().

Значительно снизить производительность графического процессора может обилие условных переходов в том или ином алгоритме [29]. Приложение GPU-MATHART_FLUX оптимизировано в смысле сокращения количества таких переходов. Кроме того, как было отмечено ранее, в алгоритме реализован подход, относящийся к классу весовых модификаций метода Монте-Карло. Такая модификация не только уменьшает число условных переходов, но и снижает дисперсию результатов.

Еще одним подводным камнем является моделирование заданного количества траекторий частиц. В случае последовательной программной реализации

метода моделирование точного количества траекторий тривиально. В случае же параллельной реализации это не так, ведь между всеми потоками, запущенными на GPU, необходима связь. Ресурсы CUDA позволяют решить эту проблему с использованием глобальной памяти и синхронизации потоков путем многократного вызова вычислительного ядра. Для повышения производительности каждый поток выполняет вычисления сразу для большого количества звеньев траектории частицы — в текущей GPU-реализации количество звеньев может варьироваться пользователем с помощью переменной NS_{GPU}. При этом если в процессе функционирования ядра траектория не оборвалась, текущее состояние фотона (координаты, направление, статистический вес) сохраняется для продолжения моделирования траектории при следующем запуске ядра.

Наиболее сложным вопросом параллельной реализации алгоритма является «сердце» метода — параллельная реализация генератора псевдослучайных чисел.

2. Параллельная реализация ГПСЧ

До последнего времени в качестве ГПСЧ в последовательных радиационных кодах MATHART, развиваемых в ИОА СО РАН, использовался предложенный еще в 1981 г. генератор Д. Кнута [30] с периодом 2^{55}, в основе которого лежит разностный метод. Усложнение разрабатываемых алгоритмов диктует необходимость использования более эффективных генераторов, подходящих, в том числе, для реализации в параллельных системах.

В приложении GPU-MATHART_FLUX в качестве ГПСЧ использован MWC-генератор (Multiply-With-Carry) [31] из семейства классических линейных конгруэнтных датчиков, применяемый также в [17, 32]. Алгоритм MWC изобретен для генерации последовательностей псевдослучайных целых чисел, построенных на основе исходного множества, включающего от двух до многих тысяч случайно выбранных начальных значений. В своей простейшей форме MWC-генератор использует аналогичную для линейного конгруэнтного генератора формулу и может быть представлен в виде

$$x_{n+1} = (x_n \times a + c) \bmod b,$$

где x — последовательность случайных чисел; a — некоторый множитель; mod — бинарная операция взятия остатка от деления на некоторую константу b; c — приращение или величина переноса от предыдущей операции mod; в отличие от линейного конгруэнтного генератора параметр c изменяется на каждой итерации.

MWC-датчик не требует большого расхода памяти, вызывает простую компьютерную целочисленную арифметику и приводит к быстрой генерации последовательностей псевдослучайных чисел с огромными периодами в диапазоне от 2^{60} до $2^{2000000}$. В отличие от классических линейных конгруэнтных генераторов MWC-датчик позволяет генерировать последовательности с большей длиной цикла при сравнимом количестве циклов работы процессора. MWC-генератор успешно прошёл статистические тесты DIEHARD [33].

При параллельной реализации генератора начальные значения множителя a определяются из заранее рассчитанного набора целых чисел, удовлетворяющих равенствам вероятностного теста Миллера–Рабина, которые выполняются для простых чисел [34, 35]. Инициализация генератора осуществляется путем задания начальных значений для x и c в пределах определенных ограничений [31]. В качестве инициализирующей константы SEED или начального состояния датчика используется текущее системное время. Эта константа применяется для инициализации первого MWC-генератора. Полученная в результате его работы последовательность псевдослучайных чисел, в свою очередь, будет являться инициализирующей для остальных генераторов. В текущей реализации приложения GPU-MATHART_FLUX все потоки вычислительного ядра работают с одной последовательностью псевдослучайных чисел, но при этом каждый поток использует свои элементы этой последовательности. Кроме того, работа с MWC-генератором организована таким образом, что при многократном последовательном запуске одного и того же численного эксперимента моделируемые траектории частиц будут различными.

Преимущество в быстродействии MWC-датчика перед рядом других ГПСЧ при моделировании $N \in \{0; 5 \cdot 10^7\}$ псевдослучайных чисел проиллюстрировано на рис. 2. В сравнении использованы последовательные CPU-реализации (встроенный в компилятор языка Си и вызываемый функцией *rand*() стандартный датчик RAND, генератор Д. Кнута RAN3 [30], генератор Вихрь Мерсенна MT_CPU [36], MWC-генератор MWC_CPU) и параллельные GPU-реализации (генератор Вихрь Мерсенна MT_GPU [36], MWC-генератор MWC_GPU) датчиков.

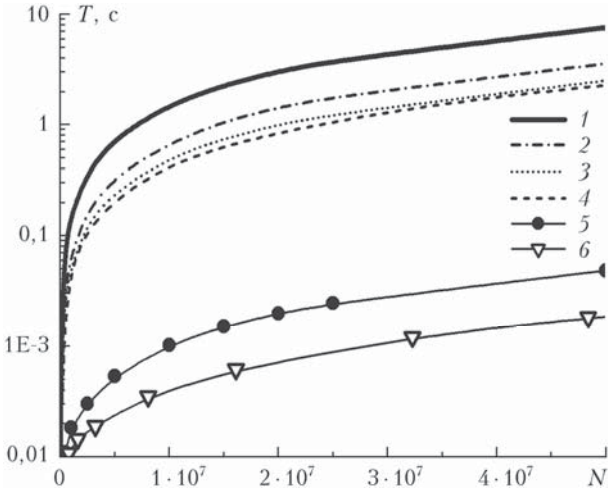


Рис. 2. Время моделирования $N \in \{0; 5 \cdot 10^7\}$ распределенных равномерно на интервале (0, 1) псевдослучайных чисел, показанное датчиками: RAND (*1*), RAN3 (*2*), MT_CPU (*3*), MWC_CPU (*4*), MT_GPU (*5*), MWC_GPU (*6*) (на базе процессоров CPU Intel Core i7-6700K и GPU NVIDIA GeForce GTX 1080)

Рис. 2 демонстрирует ярко выраженный тренд увеличения времени расчетов с увеличением количества псевдослучайных чисел. Это относится, главным образом, к генераторам, реализованным для последовательного исполнения на CPU. Наибольшее время моделирования у датчиков RAND и RAN3. Последовательные реализации MT_CPU и MWC_CPU обеспечили сравнительно близкое и лучшее среди CPU-реализаций время моделирования 50 млн чисел, равное 2,4 и 2,3 с соответственно. Показатели совершенно иного порядка имеют место для GPU-реализаций этих генераторов: например, на моделирование такого количества чисел с помощью MWC-генератора уходит лишь 0,019 с. Таким образом, параллельный MWC-алгоритм демонстрирует наивысшее быстродействие, в 120 раз большее, чем в своей последовательной реализации.

3. Результаты тестирования

Тестирование кода GPU-MATHART_FLUX выполнено с помощью серии численных экспериментов. Моделирование распространения солнечного излучения осуществляется в оптически плотной среде в спектральных каналах 0,44; 0,76 и 0,94 мкм.

Используемые в расчетах данные соответствуют модели оптических характеристик аэрозоля и облаков OPAC (Optical Properties of Aerosol and Clouds) [38]. Оптические свойства аэрозоля соответствуют типу Continental Average. Рассмотрены три типа облачности — кучевая (*Cumulus*), слоистая (*Stratus Opacus*) и перистая (*Cirrus*), оптические характеристики (индикатриса и альбедо однократного рассеяния) которых определяются типами Cumulus Continental, Stratus Continental и Cirrus3 из [38] соответственно. Оптическая толщина облаков (OTO) и аэрозольная оптическая толща (AOT) варьировались в зависимости от цели задачи.

Вычислительная система I (*C*I), с помощью которой были получены анализируемые далее результаты, состояла из четырехъядерного процессора шестого поколения Intel Core i7-6700K 4,0 ГГц и видеокарты NVIDIA GeForce GTX 1080 (первой модели высокого класса, основанной на графическом процессоре новой архитектуры Pascal). Ускоритель, включающий 20 потоковых мультипроцессоров и 2560 скалярных процессоров, демонстрирует вычислительную мощность почти 9 TFLOPS в операциях с одинарной точностью. Уровень вычислительных возможностей CC видеокарты 6.1. В целях сравнительного тестирования была использована видеокарта более низкого уровня NVIDIA GeForce GTX 950M (архитектура Maxwell) с CC 5.0, установленная на ПК с двухъядерным процессором шестого поколения Intel Core i7-6500U 2,5 ГГц (вычислительная система II (*C*II)).

Основными критериями эффективности применения GPU являются скорость вычислений, а также ускорение решения задачи на GPU по сравнению с традиционными вычислениями на процессоре общего назначения. Для характеристики такого ускорения введем безразмерную величину A_T, равную отношению времени расчета T_{CPU} по CPU-алгоритму ко времени расчета T_{GPU} по GPU-алгоритму. Отметим, что функции отсечки времени в том и другом алгоритмах реализованы непосредственно до и после блока моделирования траекторий фотонов.

Задача численного моделирования распространения солнечного излучения в атмосфере сопряжена с большим количеством входных параметров, необходимых для ее решения. Прогноз того, какой будет производительность параллельного кода при вариации какого-либо из этих параметров, не является тривиальным и требует проведения соответствующих исследований.

Первая группа тестов ограничена двумя простыми моделями: рассматриваются плоские однородные аэрозольный (Continental Average) и облачный (Cumulus Continental) слои высотой 1 км. Если иное не оговорено особо, солнце находится в зените, а отражение от нижней границы слоя осуществляется в соответствии с альбедо черного тела. С целью упрощения пренебрегаем молекулярным поглощением и рассеянием. На этих простых примерах мы исследуем, как влияют на скорость GPU-вычислений различные параметры эксперимента (альбедо однократного рассеяния частиц, альбедо подстилающей поверхности, оптическая плотность слоя, положение солнца, количество моделируемых траекторий фотонов), а также конфигурация вычислительного ядра.

Предполагается, что на верхнюю границу слоя падает единичный поток солнечного излучения. Представленные далее результаты получены на длине волны 0,44 мкм. Количество моделируемых траекторий равно 10 млн, что является более чем достаточным для обеспечения высокой точности расчетов. Погрешности расчетов, выполненных тем и другим алгоритмом, сопоставимы между собой и не превышают 0,5%.

Один из факторов, определяющих быстродействие GPU-кода — это выбор конфигурации ядра, состоящий в определении значений параметров параллельного исполнения N_b (число параллельных блоков) и N_t (число нитей в одном блоке). Для определения оптимальной конфигурации требуется сначала выбрать размер блока с пиковыми значениями производительности, а затем выбрать число блоков, для которого это пиковое значение достигается. Чаще всего это задача решается экспериментально — путем перебора возможных конфигураций вычислительного ядра. Разработка аналитического подхода определения оптимальной конфигурации и эффективного использования имеющихся ресурсов требует отдельного рассмотрения и здесь не обсуждается.

На базе расчетов потоков нисходящей и восходящей солнечной радиации в облачном слое с оптической толщиной 50 была построена матрица производительности кода в зависимости от N_b и N_t. На рис. 3 представлено время вычислений T при разных N_b и N_t для вычислительных систем I и II соответственно. Как видно из рисунка, при увеличении числа блоков, а значит, и вычислительной нагрузки производительность ядра некоторое время растет, затем происходит резкий спад. При этом пиковые

значения производительности для разных размеров блока различаются. Как для системы I, так и системы II оптимальная комбинация параметров $N_b = 96$ и $N_t = 128$ с общим количеством потоков 12288. Эта конфигурация ($K1$) и была использована во всех расчетах в рамках однослойной модели.

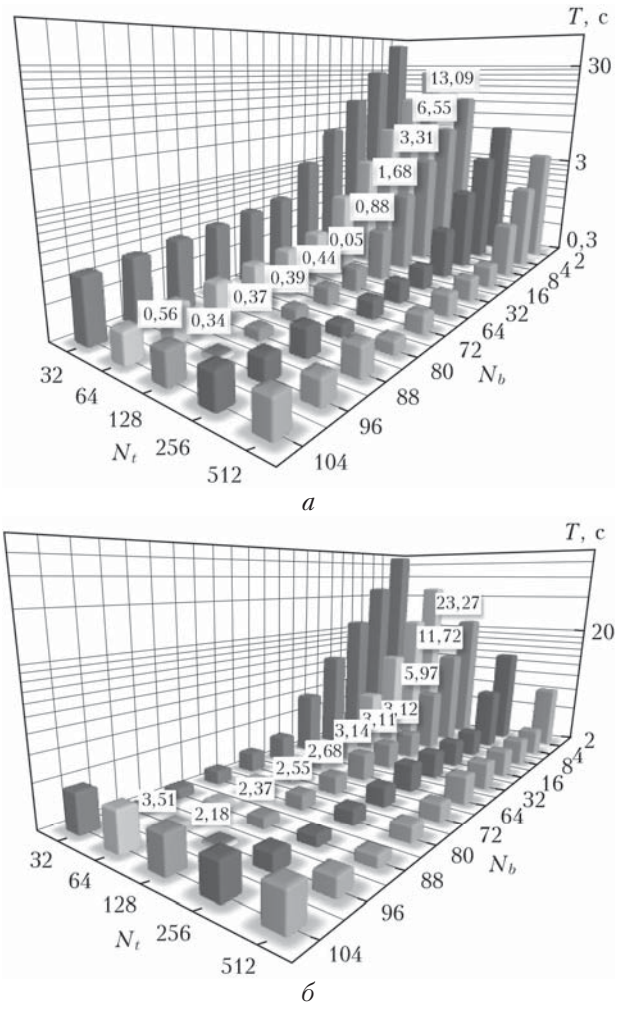


Рис. 3. Экспериментальная зависимость времени работы ядра от числа параллельных блоков и числа нитей в одном блоке для вычислительных систем I (*а*) и II (*б*)

Рассмотрим диапазон изменения величины ускорения в зависимости от оптической толщины облачного слоя. На рис. 4 сплошной кривой обозначено время, затрачиваемое на последовательный CPU-расчет при изменении ОТО от 5 до 200 (величина ОТО = 200 является гипотетической и может не отражать реальной действительности; служит неким пределом для демонстрации возможностей используемой технологии и разработанного кода для решения задач в оптически очень плотных средах). При максимальном ОТО время расчета потоков составляет около 12 мин, тогда как использование технологии CUDA позволяет провести аналогичные вычисления всего за 1,3 с. Наблюдается устойчивая тенденция роста ускорения A_T от ~260 до 560 с увеличением оптической толщины слоя от 5 до 200 соответственно. Использование другой конфигурации ядра, например $N_b = 64$ и $N_t = 256$ с общим количеством потоков 16384 ($K2$), снижает эффективность параллельной реализации: максимум величины A_T в этом случае равен 400 (на рисунке не показано).

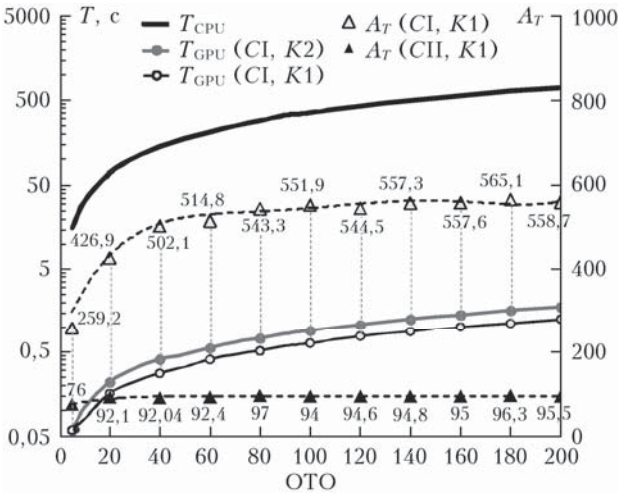


Рис. 4. Время расчета T потоков с помощью CPU- и GPU-кодов в облачном слое при разной оптической толщине облака и ускорение A_T, обеспечиваемое GPU-алгоритмом

Эффективность параллельного кода падает при использовании менее мощной видеокарты NVIDIA GeForce GTX 560M (вычислительная система II) с той же конфигурацией ядра: пиковое значение ускорения не превышает 97 (см. рис. 4).

Следует отметить существование некоторого предела величины A_T. В последнем случае этот предел достигается уже при сравнительно небольших ОТО и далее наблюдаются лишь колебания A_T в его окрестности. В первом же случае предел достигается с ростом ОТО постепенно (см. рис. 4).

С ростом зенитного угла солнца (ЗУС) от 0 («зенит») до 80° (углы более 80° требуют перехода к сферической модели атмосферы и здесь не рассматриваются) величина ускорения падает (таблица). При этом чем более прозрачна среда, тем стремительнее это падение. В слое с ОТО = 10 главная причина такого поведения A_T — снижение времени работы CPU-программы с ростом ЗУС на фоне практически неизменного времени работы GPU-кода при любых зенитных углах. При ОТО = 50 поведение A_T определяется уже взаимным изменением T_{CPU} и T_{GPU}.

Ускорение A_T в зависимости от зенитного угла солнца при разных оптических толщинах облачного слоя

ОТО	ЗУС, град										
	0	10	20	30	40	50	60	65	70	75	80
10	406,3	360,0	417,5	396,3	386,3	372,5	342,5	325,0	303,8	280,0	247,5
50	515,0	510,0	513,0	498,8	479,4	417,3	466,3	458,8	455,2	469,5	430,0

Важно отметить, что в отдельных численных экспериментах (оптически плотная облачность, сумерки и т.д.) выбранного количества моделируемых траекторий фотонов N_{ph} недостаточно для обеспечения приемлемой точности вычислений. На примере расчетов в оптически плотном облачном слое (ОТО = 50) выявлено, что с увеличением количества траекторий N_{ph} от 2,5 до 50 млн время выполнения расчета с помощью CPU-программы T_{CPU} возросло до 14,5 мин, увеличившись примерно в 20 раз. При этом выигрыш GPU-кода вырос с 485 до 550, обеспечив конечное время расчета 1,6 с при погрешности вычислений не более 0,05%. Таким образом, достижение высокой статистической достоверности результатов расчетов с помощью GPU-кода посредством существенного увеличения N_{ph} может быть выполнено без ощутимых временных потерь.

Численные эксперименты с облачным слоем показали, что с уменьшением оптической плотности области моделирования показатели ускорения параллельного алгоритма по отношению к последовательному CPU-алгоритму падают, хотя и остаются на приемлемо высоком уровне. Тестирование приложения GPU-MATHART_FLUX на аэрозольном слое демонстрирует падение ускорения A_T от 465 до 76 при изменении АОТ от 10 до 1. Снижение A_T обусловлено, главным образом, замедлением работы CPU-кода с ростом АОТ на фоне почти неизменного времени работы GPU-кода, которое в данном случае составляет примерно 0,06 с. Ускорение A_T в среде с АОТ 0,75; 0,5 и 0,25 составило 50, 37 и 25 соответственно.

Вариации значений альбедо однократного рассеяния аэрозоля (от 0,6 до 1) и альбедо подстилающей поверхности (от 0 до 1) не оказывают значимого влияния на время GPU-вычислений. При этом с ростом длины волны — в отсутствие молекулярного поглощения — показатели ускорения несколько снижаются: в рамках рассматриваемой оптической модели коэффициенты рассеяния рэлеевских и аэрозольных частиц с увеличением длины волны существенно уменьшаются.

Таким образом, тесты свидетельствуют о том, что оптически плотный однородный слой с минимальным набором описывающих его параметров является средой, обеспечивающей высокие показатели ускорения относительно последовательной CPU-реализации алгоритма метода Монте-Карло. Переход к многослойной модели сопряжен с введением дополнительных параметров и существенным увеличением размерности задачи, что обусловливает дополнительное расходование памяти и снижение эффективности распараллеливания. Предварительные расчеты потоков в аэрозольном слое с АОТ = 1 показали, что учет вертикальной стратификации атмосферы, влекущий за собой выполнение дополнительного цикла операций и усложнение расчетов в целом, приводит к снижению скорости вычислений с помощью GPU-кода примерно в 2,5 раза и в 1,5 раза — с помощью CPU-реализации. Этот результат был получен путем разбиения однородного слоя на 45 подслоев с одинаковыми оптическими свойствами и сравнения результатов моделирования с данными, полученными для слоя без разбиения.

В следующей группе тестов свойства среды описываются вертикальными профилями оптических характеристик, определяющими многослойную аэрозольно-молекулярную модель атмосферы. Модель разделена на 45 плоскопараллельных слоев. Коэффициенты рэлеевского рассеяния рассчитываются на каждом уровне в соответствии с вертикальными профилями температуры и давления для летних условий умеренных широт [39]. АОТ в спектральном канале 0,44 мкм составляет 0,67.

Для проведения расчетов в рамках этой модели, в том числе при наличии облачности, также была найдена оптимальная конфигурация вычислительного ядра с параметрами N_b и N_t, равными 80 и 256 соответственно ($K3$).

Рассмотрим сначала результаты моделирования, выполненные без учета поглощения излучения атмосферными газами.

Наряду с количеством уровней стратификации модели атмосферы имеется еще один потенциальный фактор, определяющий производительность параллельных вычислений в рассматриваемом случае, а именно количество уровней, на которых рассчитываются потоки нисходящей и восходящей солнечной радиации, совпадающие, как правило, с уровнями стратификации. Переход от расчетов на двух уровнях к расчетам, например, на десяти уровнях приводит к увеличению T_{GPU} от 0,36 до 0,57 с, т.е. почти в 1,5 раза. Обусловлено это использованием атомарных операций, необходимых для накопления общего результата при обработке каждой пачки траекторий для каждого уровня (см. рис. 1). Как известно, атомарные операции, обеспечивающие доступ к медленной глобальной памяти GPU, длительнее обычных — они занимают примерно 50 тактов центрального процессора. Однако благодаря использованию атомарных инструкций не требуется переход в режим ядра, занимающий тысячи тактов. Поскольку в большинстве практических приложений теории переноса достаточными являются расчеты потоков только на уровнях верхней границы атмосферы и подстилающей поверхности, данный фактор не несет в себе большой угрозы производительности. В последовательной CPU-реализации алгоритма такая проблема отсутствует, что в совокупности и приводит к сниженным показателям ускорения A_T.

Кривые, характеризующие время вычислений потоков с помощью CPU- и GPU-алгоритмов в многослойной атмосфере при наличии сплошного слоя кучевой облачности, расположенного на высоте от 1 до 2 км (ОТО = 50), представлены на рис. 5. GPU-код демонстрирует быстрый рост ускорения с ростом ОТО — от 23 при оптической толщине 5 до 173,5 при оптической толщине 100. При ОТО = 200 GPU-код быстрее CPU-реализации в 255 раз.

Ускорение, достигаемое при наличии в атмосфере на высоте от 8 до 9 км сплошного слоя оптически тонкой перистой облачности, невелико и при ОТО = 1 не превышает 15, а при ОТО = 2 — ~18. Сравнительно низкие значения A_T обусловливает низкая оптическая плотность среды.

Как и в случае с однородным слоем, в многослойной модели с однослойной облачностью имеет

место аналогичная зависимость ускорения A_T от зенитного угла солнца.

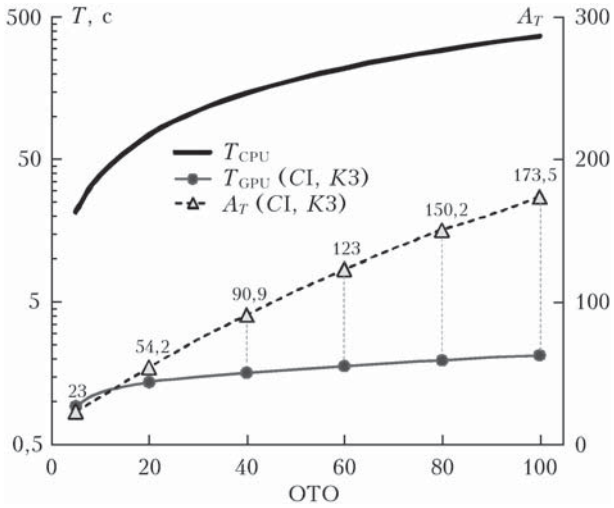


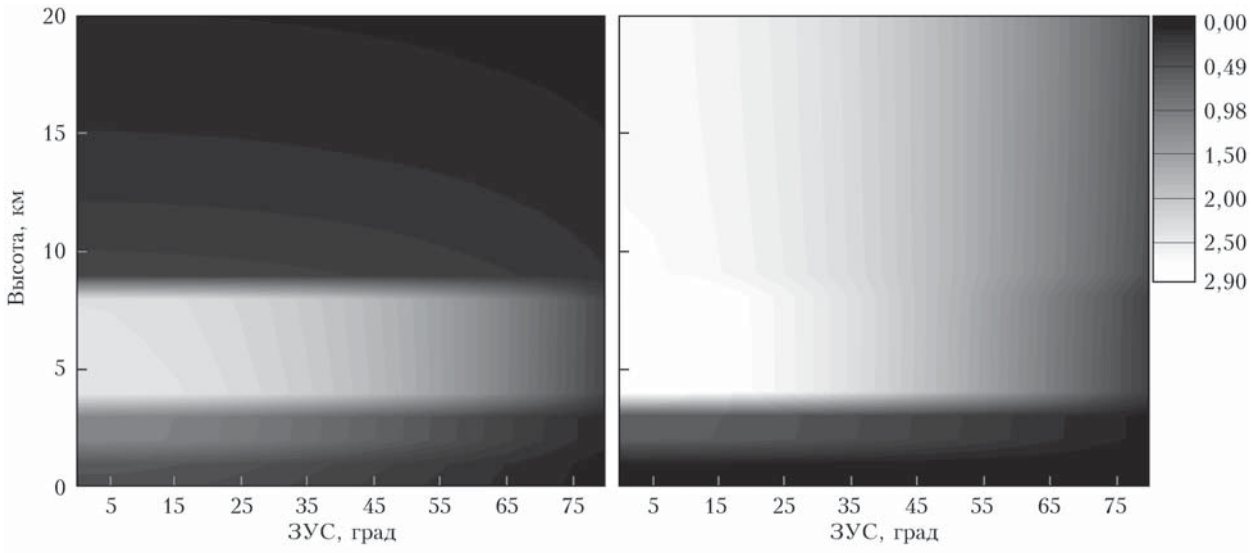
Рис. 5. Время расчета T потоков с помощью CPU- и GPU-кодов в облачной атмосфере при разной оптической толщине облака; ускорение A_T, обеспечиваемое GPU-алгоритмом

В реальной атмосфере облачные слои могут присутствовать на разных уровнях одновременно. Такая расслоенность — важная характеристика их пространственной структуры и обусловлена, как правило, атмосферными фронтами либо распадом мощных кучевых и кучево-дождевых облаков [40]. По данным [40], над территорией бывшего СССР без учета верхнего яруса в 30–50% случаев облачность однослойная, примерно в 30% случаев — двухслойная и примерно в 20% случаев зимой и 30% случаев летом — трех- или четырехслойная. Поскольку разработанное приложение GPU-MATHART_FLUX позволяет моделировать перенос излучения с учетом многослойной облачности, тесты проведены и для случая, когда облака разных типов присутствуют в атмосфере одновременно: на высоте от 1 до 2 км расположено слоистое облако (ОТО = 10), на высоте от 3 до 4 — кучевое облако (ОТО = 50), на высоте от 8 до 9 км — перистое облако (ОТО = 1). Результаты моделирования потоков нисходящей и восходящей солнечной радиации на 45 уровнях стратификации в многослойной облачности при изменении зенитного угла солнца от 0 до 80° представлены на рис. 6. С использованием графического ускорителя этот результат получен за 1 мин и 11 с, тогда как при использовании последовательной CPU-программы расчет занял 1 ч 2 мин.

Из рис. 6, *а* видно, что большая часть фотонов рассеялась в пространстве, ограниченном перистым и слоистым облаками, обеспечив максимальные значения потока нисходящей радиации, убывающими с ростом ЗУС. В данном случае имеет место эффект взаимного переотражения излучения облаками. Для отраженного потока слои кучевой и слоистой облачности стали блокирующими для дальнейшего распространения излучения к поверхности Земли, поэтому наибольшие значения потока, возрастающие с уменьшением зенитного угла солнца, наблюдаются на высотах более 3–4 км.

Далее рассмотрим, как меняется производительность параллельного GPU-кода в результате ввода в модель селективного молекулярного поглощения излучения.

Учет молекулярного поглощения предполагает появление в программном коде новых параметров — эффективных коэффициентов поглощения и коэффициентов гауссовских квадратур, и как следствие — больший расход памяти, увеличение сложности алгоритма, а значит, и времени, затрачиваемого на расчет.



а *б*

Рис. 6. Потоки нисходящей (*а*) и восходящей (*б*) солнечной радиации (отн. ед.) в аэрозольно-молекулярной атмосфере с многослойной облачностью

Эффективные коэффициенты поглощения вычислялись на основе спектроскопической базы параметров спектральных линий HITRAN2008 с учетом реальных метеопараметров атмосферы и профилей концентрации атмосферных газов (озон O_3, кислород O_2, углекислый газ CO_2, метан CH_4, водяной пар H_2O и др.) модели AFGL [41] методом k-распределений, позволяющим заменить быстро осциллирующий коэффициент поглощения атмосферных газов на монотонную функцию и представить функцию пропускания в виде конечного ряда экспонент [42, 43]. Для параметризации функции пропускания использовались 10 квадратур Гаусса.

Рассматривались два спектральных канала шириной 0,25 мкм: 0,76 и 0,94 мкм. Первый интервал — это полоса кислорода O_2, широко используемая в дистанционном спутниковом зондировании атмосферы, в том числе для определения высоты верхней границы облаков [44], параметров облачности [45], а также характеристик аэрозольного состояния атмосферы [46, 47]; второй интервал — одна из наиболее сильных в ближней ИК-области спектра полос поглощения водяного пара H_2O.

Результаты тестирования показали, что учет селективного молекулярного поглощения приводит к заметному увеличению времени CPU-вычислений даже в безоблачной атмосфере: в полосах умеренного и сильного поглощения значения некоторых из эффективных коэффициентов молекулярного поглощения существенно возрастают. Время расчета спектральных потоков T_{CPU} тем больше, чем сильнее поглощение: в спектральном канале 0,76 мкм T_{CPU} составило 47,3 с при AOT = 0,31, тогда как в канале 0,94 мкм — 51,5 с при AOT = 0,23. Производительность же параллельного кода в рамках экспериментов в безоблачной атмосфере, наоборот, существенно выросла по сравнению со случаем, когда молекулярное поглощение отсутствовало. Время расчета T_{GPU} спектральных потоков в том и другом каналах составило 0,55 и 0,44 с соответственно, что меньше T_{CPU} в 86 и 117 раз.

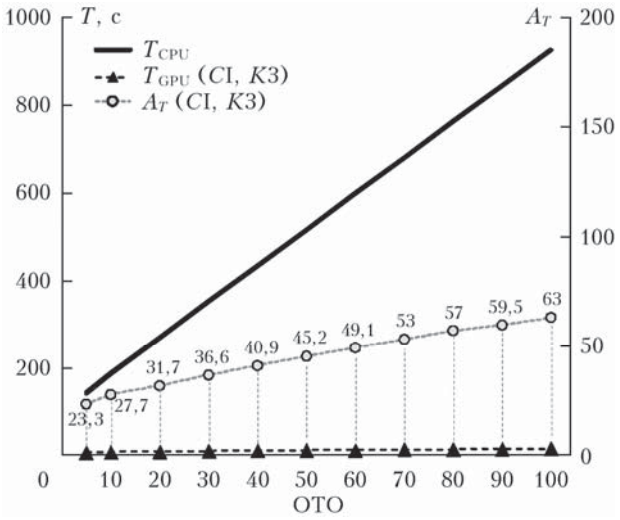


Рис. 7. Время расчета T потоков с помощью CPU- и GPU-кодов в облачной атмосфере с учетом селективного молекулярного поглощения при разной оптической толщине облака; ускорение A_T, обеспечиваемое GPU-алгоритмом

При появлении сплошной однослойной облачности ситуация несколько меняется. Существенно возросшее количество актов взаимодействия фотона со средой, при каждом из которых осуществляется пересчет искомых оценок по количеству гауссовских квадратур, не позволяет GPU-коду достигнуть высоких значений A_T. С увеличением оптической толщины облаков от 5 до 100 величина A_T меняется незначительно — от 29 до 63 (рис. 7). При OTO = 200 величина ускорения не превышает 87.

Следует отметить, что с уменьшением числа квадратур эффективность параллельного кода в облачной атмосфере будет расти, однако точность расчетов будет снижаться.

Заключение

Рассмотрены основные принципы параллельной реализации метода Монте-Карло на графических процессорах NVIDIA, совместимых с технологией CUDA. На примере расчетов спектральных потоков солнечной радиаций в безоблачной и облачной атмосфере показано, что разработанный GPU-алгоритм позволяет эффективно использовать параллельную архитектуру графического ускорителя.

Анализ быстродействия GPU-кода выполнен путем оценки времени моделирования распространения излучения в плоском однородном слое разной оптической плотности, а также в вертикально-неоднородной модели атмосферы, в том числе в присутствие однослойной и многослойной облачности. Проведено исследование чувствительности ускорения A_T, обеспечиваемого параллельным GPU-алгоритмом относительно последовательной CPU-реализации, к оптико-геометрическим параметрам атмосферы.

Выявлено, что в отсутствие молекулярного поглощения ускорение A_T растет с увеличением оптической плотности среды и с уменьшением зенитного угла солнца. Вариации альбедо однократного рассеяния аэрозоля и альбедо подстилающей поверхности не оказывают значимого влияния на время GPU-вычислений. Учет селективного молекулярного поглощения приводит к росту ускорения в безоблачной атмосфере и снижению при появлении облачности.

Производительность GPU-кода зависит также от выбора конфигурации вычислительного ядра и от производительности самого графического процессора.

Несмотря на очевидный параллелизм задачи, оптимизация кода на графическом процессоре NVIDIA архитектуры CUDA оказалась весьма сложным процессом. В частности, для того чтобы параллельные потоки могли одновременно вносить изменения в общие результирующие матрицы, необходимо многократное использование атомарных инструкций для доступа к медленной глобальной памяти GPU. В силу стохастической природы метода Монте-Карло переход к его параллельной программной реализации должен быть обеспечен процедурой параллельной генерации независимых последовательностей псевдослучайных чисел высокого качества. Другой проблемой

является параллельная реализация подхода к моделированию точного количества траекторий.

Переход к использованию GPU сократил время моделирования распространения солнечного излучения в среднем на 2 порядка по сравнению с CPU общего назначения, убедительно показывая преимущество в скорости параллельных алгоритмов перед их классическими последовательными реализациями. Достигнутые показатели не являются предельными. Стремительное развитие графических процессоров и сопутствующих программных технологий будет способствовать росту производительности параллельного GPU-алгоритма.

Дальнейшее усовершенствование предложенного кода и реализация максимального вычислительного потенциала видеокарты возможны также за счет: а) разработки более гибкого алгоритма работы с атомарными операциями, б) более широкого использования специальных областей видеопамяти, в) одновременной эксплуатации нескольких видеокарт в одном ПК.

В отличие от CPU-реализаций радиационных кодов, ориентированных как на отдельные ПК, так и на дорогостоящие кластерные вычислительные системы, разработка высокопроизводительного кода на базе графических ускорителей NVIDIA существенно снижает себестоимость затрат на получение новых результатов, позволяя решать прорывные задачи науки и техники, требующие больших объемов вычислений, в короткий срок, снимает необходимость использования разного рода приближений и параметризаций, созданных ранее для увеличения быстродействия алгоритмов, позволяет перевести процессы обработки и имитации данных измерений в режим, близкий к реальному времени.

Работа выполнена при финансовой поддержке РФФИ (проект № 16-31-60057 мол_а_дк) и Министерства образования и науки (грант Президента РФ № МК-5381.2016.5).

1. *Dubovik O., Lapyonok T., Litvinov P., Herman M., Fuertes D., Ducos F., Lopatin A., Chaikovsky A., Torres B., Derimian Y., Huang X., Aspetsberger M., Federspiel C.* GRASP: A versatile algorithm for characterizing the atmosphere // SPIE: Newsroom. 2014. DOI: 10.1117/2.1201408.005558.
2. *Ленобль Ж.* Перенос радиации в рассеивающих и поглощающих атмосферах. Л.: Гидрометеоиздат, 1990. 264 с.
3. *Марчук Г.И., Михайлов Г.А., Назаралиев М.А., Дарбинян Р.А., Каргин Б.А., Елепов Б.С.* Метод Монте-Карло в атмосферной оптике. Новосибирск: Наука, 1976. 280 с.
4. *Marshak A., Davis A.B.* 3D radiative transfer in cloudy atmospheres. Berlin: Springer, 2005. 686 p.
5. *Moore G.* Litography and the future of Moore's law // Proc. SPIE. 1995. V. 2437. P. 2–17.
6. *Kirkby D.R., Delpy D.T.* Parallel operation of Monte Carlo simulations on a diverse network of computers // Phys. Med. Biol. 1997. V. 42, N 6. P. 1203–1208.
7. *Colasanti A., Guida G., Kisslinger A., Liuzzi R., Quarto M., Riccio P., Roberti G., Villani F.* Multiple processor version of a Monte Carlo code for photon transport in turbid media // Comput. Phys. Commun. 2000. V. 132, N 1–2. P. 84–93.
8. *Кожевникова А.В., Тарасенков М.В., Белов В.В.* Параллельные вычисления при решении задач восстановления коэффициента отражения земной поверхности по спутниковым данным // Оптика атмосф. и океана. 2013. Т. 26, № 2. С. 172–174; *Kozhevnikova A.V., Tarasenkov M.V., Belov V.V.* Parallel computations for solving problem of the reconstruction of the reflection coefficient of the Earth's surface by satellite data // Atmos. Ocean. Opt. 2013. V. 26, N 4. P. 326–328.
9. *Глинский Б.М., Родионов А.С., Марченко М.А., Подкорытов Д.И., Винс Д.В.* Агентно-ориентированный подход к имитационному моделированию супер ЭВМ экзафлопсной производительности в приложении к распределенному статистическому моделированию // Вестн. ЮУрГУ. 2012. № 18(277), вып. 12. С. 94–99.
10. *Волков К.Н., Дерюгин Ю.Н., Емельянов В.Н., Карпенко А.Г., Козелков А.С., Тетерина И.В.* Методы ускорения газодинамических расчетов на неструктурированных сетках. М.: ФИЗМАТЛИТ, 2014. 536 с.
11. *Боресков А.В., Харламов А.А., Марковский Н.Д., Микушин Д.Н., Мортиков Е.В., Мыльцев А.А., Сахарных Н.А., Фролов В.А.* Параллельные вычисления на GPU: архитектура и программная модель CUDA. М.: Изд-во Моск. ун-та, 2012. 336 с.
12. *Zhu C., Liu Q.* Review of Monte Carlo modeling of light transport in tissues // J. Biomed. Opt. 2013. V. 18, N 5. P. 050902-1–050902-12.
13. *Фикс И.И.* Использование графических процессоров для решения задачи распространения света в диффузионной флуоресцентной томографии методом Монте-Карло // Вестн. Нижегородск. ун-та им. Н.И. Лобачевского. 2011. № 4(1). С. 190–195.
14. *Кириллин М.Ю., Фикс И.И., Катичев А.Р., Горшков А.В., Гергель В.П.* Высокопроизводительные вычисления для задач оптической биомедицинской диагностики // Суперкомпьютерные технологии в науке, образовании и промышленности / под ред. В.А. Садовничего, Г.И. Савина, Вл.В. Воеводина. Вып. 3. М.: Изд-во Моск. ун-та, 2012. 232 с.
15. *Петров Д.А.* Моделирование оптических методов биомедицинской диагностики // Концепт. 2016. Т. 11. С. 2851–2855.
16. *Alerstam E., Lo W.C.Y., Han T.D., Rose J., Anderson-Engels S., Lilge L.* Next-generation acceleration and code optimization for light transport in turbid media using GPUs // Biomed. Opt. Express. 2010. V. 1, N 2. P. 658–675.
17. *Alerstam E., Svensson T., Anderson-Engels S.* Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration // J. Biomed. Opt. 2008. V. 13, N 6. P. 060504-1–060504-3.
18. *Cai F., He S.* Using graphics processing units to accelerate perturbation Monte Carlo simulation in a turbid medium // J. Biomed. Opt. 2012. V. 17, N 4. P. 040502-1–040502-3.
19. *Martinsen P., Blaschke J., Künnenmeyer R., Jordan R.* Accelerating Monte Carlo simulations with an NVIDIA graphics processor // Comput. Phys. Commun. 2009. V. 180, N 10. P. 1983–1989.
20. *Fang Q., Boas D.A.* Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units // Opt. Express. 2009. V. 17, N 22. P. 20178–20190.
21. *Ren N., Liang J., Qu X., Li J., Lu B., Tian J.* GPU-based Monte Carlo simulation for light propagation in complex heterogeneous tissues // Opt. Express. 2010. V. 18, N 7. P. 6811–6823.
22. *Efremenko D.S., Loyola D.G., Doicu A., Spurr R.J.D.* Multi-core-CPU and GPU-accelerated radiative transfer

models based on the discrete ordinate method // Comput. Phys. Commun. 2014. V. 185, N 12. P. 3079–3089.
23. *Ramon D., Steinmetz F., Compiegne M., Jolivet D.* Massively parallel Monte-Carlo radiative transfer code on a desktop PC. URL: http://www-loa.univ-lille1.fr/workshops/Trattoria-2015/documents/posters/Didier_Ramon.pdf (last access: 1.10.2017).
24. *Clough S.A., Iacono M.J., Moncet J.L.* Line-by-line calculations of atmospheric fluxes and cooling rates: Application to water vapor // J. Geophys. Res. D. 1992. V. 97. P. 16519–16535.
25. *NVIDIA* official page. URL: www.nvidia.ru (lass access: 01.10.2017).
26. *Русскова Т.В., Журавлева Т.Б.* Оптимизация последовательного программного кода для моделирования переноса солнечного излучения в вертикально-неоднородной среде // Оптика атмосф. и океана. 2016. Т. 29, № 10. С. 836–842; *Russkova T.V., Zhuravleva T.B.* Optimization of sequential code for simulation of solar radiation transfer in a vertically heterogeneous environment // Atmos. Ocean. Opt. 2017. V. 30, N 2. P. 169–175.
27. *Франк-Каменецкий А.Д.* Моделирование траекторий нейтронов при расчете реакторов методом Монте-Карло. М.: Атомиздат, 1978. 93 с.
28. *Сандерс Дж., Кэндрот Э.* Технология CUDA в примерах: введение в программирование графических процессоров. М.: ДМК Пресс, 2015. 232 с.
29. *Жуковский М.Е., Усков Р.В.* Математическое моделирование радиационной эмиссии электронов на гибридных суперкомпьютерах // Вычислительные методы и программирование. 2012. Т. 13. С. 271–279.
30. *Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.* Numerical Recipes in Fortran 77: The Art of Scientific Computing. Volume 1 of Fortran Numerical Recipes. Cambridge University Press, 1986. 1003 pp.
31. *Marsaglia G.* Random Number Generators // J. Mod. Appl. Stat. Methods. 2003. V. 2, N 1. P. 2–13.
32. *Lee A., Yau C., Giles M.B., Doucet A., Holmes C.C.* On the utility of graphic cards to performe massively parallel simulation of advanced Monte Carlo methods // J. Comput. Graph. Stat. 2010. V. 19, N 4. P. 769–789.
33. *Marsaglia G.* Diehard battery of tests of randomness. The Marsaglia random number CD ROM. Department of Statistics, Florida State University, 1995.
34. *Miller G.L.* Riemann's hypothesis and tests for primarily // J. Comput. Syst. Sci. 1976. V. 13. P. 300–317.
35. *Rabin M.O.* Probabilistic algorithm for testing primarily // J. Number Theory. 1980. V. 20, N 1. P. 128–138.
36. *Matsumoto M., Nishimura T.* Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator // ACM Trans. Model. Comput. Sim. 1998. V. 8, N 1. P. 3–30.
37. *Matsumoto M., Nishimura T.* Dynamic creation of pseudorandom number generators // Monte Carlo and Quasi-Monte Carlo Methods. Springer, 2000. P. 56–69.
38. *Hess M., Koepke P., Schult I.* Optical properties of aerosols and clouds: The software package OPAC // Bull. Am. Meteorol. Soc. 1998. V. 79, N 5. P. 831–844.
39. *Комаров В.С., Ломакина Н.Я.* Статистические модели пограничного слоя атмосферы Западной Сибири. Томск: Изд-во ИОА СО РАН, 2008. 222 с.
40. *Мазин И.П., Хргиан А.Х., Имянинов И.М.* Облака и облачная атмосфера. Л.: Гидрометиздат, 1989. 647 с.
41. *Anderson G.P., Clough S.A., Kneizys F.X., Chetwynd J.H., Shettle E.P.* AFGL Atmospheric Constituent Profiles (0–120 km). Air Force Geophysics Laboratory, 1986. 46 p.
42. *Фирсов К.М., Смирнов А.Б.* Представление функций пропускания рядом экспонент // Оптика атмосф. и океана. 1995. Т. 8, № 8. С. 1248–1252.
43. *Творогов С.Д.* Некоторые аспекты задачи о представлении функции поглощения рядом экспонент // Оптика атмосф. и океана. 1994. Т. 7, № 3. С. 315–326.
44. *Fischer J., Grassl H.* Detection of cloud top height from backscattered radiances within the oxygen *A* band. Part 1. Theoretical study // J. Appl. Meteorol. 1991. N 30. P. 1245–1259.
45. *Koelemeijer R.B.A., Stammes P., Hovenier J.W., de Haan J.F.* A fast method for retrieval of cloud parameters using oxygen *A* band measurements from the Global Ozone Monitoring Experiment // J. Geophys. Res. 2001. V. 106. P. 3475–3490.
46. *Бадаев В.В., Малкевич М.С., Низик Б., Циммерман Г.* Определение оптических параметров земной поверхности, океана и атмосферы со спутников «Интеркосмос 20 и 21» // Исслед. Земли из космоса. 1985. № 5. С. 18–29.
47. *Timofeyev Yu.M., Vasilyev A.V., Rozanov V.V.* Information content of the spectral measurement of the 0.76 μm O_2 outgoing radiation with respect to the vertical aerosol optical properties // Adv. Space Res. 1995. V. 16, N 10. P. 91–94.

T.V. Russkova. **Monte Carlo simulation of solar radiative transfer in the cloudy atmosphere using graphics processor and NVIDIA CUDA technology.**

Issues about improving the performance of Monte Carlo numerical simulation of light transport in the Earth's atmosphere by moving from consecutive calculations to parallel ones are discussed. A new parallel algorithm oriented to a computing system with a graphics processor that supports the NVIDIA CUDA technology is suggested. The efficiency of parallelization is analyzed on the basis of calculating the fluxes of downward and upward solar radiation in both vertically homogeneous and heterogeneous models of the atmosphere. The results of approbation of the new code under various atmospheric conditions including continuous single-layered and multilayered clouds and selective molecular absorption are presented. The results of testing the code using video cards with different compute capability are analyzed. It is shown that the changeover of computing from conventional PCs to the architecture of graphics processors gives more than a hundredfold gain in performance and fully reveals the capabilities of the technology used.