Программирование для нескольких GPU на примере задачи распространения света в многослойной среде

Багаутдинов Т.А.[1], Гергель В.П.[1], Горшков А.В.[1], Фикс И.И.[1,2], Кириллин М.Ю.[1,2]
[1]*Нижегородский госуниверситет им. Н.И. Лобачевского*
[2]*Институт прикладной физики РАН*

Введение

В настоящее время лазерные методы получили широкое распространение в бесконтактной неразрушающей диагностике внутренней структуры различных оптически неоднородных объектов, в частности, они находят применение в медицине, биофизике, науках о материалах, физике атмосферы, и других областях.

Для повышения эффективности современных методов лазерной диагностики, а также для разработки новых методов, необходимо подробное изучение особенностей процесса распространения света в различных средах, включая биоткани. Однако решение данной задачи затруднено тем, что на данный момент не существует точной теории для описания распространения света в структурно неоднородных средах, а экспериментальные исследования осложнены трудностями поддержания постоянства их структурно-динамических параметров. В связи с этим все большую роль приобретает компьютерное моделирование этого процесса (см., например, [1]). Данный подход позволяет более тщательно изучить особенности процесса распространения лазерного пучка в модельных средах, а также исследовать зависимость получаемых результатов от различных параметров измерительной системы и исследуемого объекта, что бывает весьма затруднительно в эксперименте.

Постановка задачи

В рамках данной работы ставится задача моделирования распространения света в многослойной среде. При этом считаем, что задача решается в трехмерном пространстве, а среда состоит из набора плоскопараллельных слоев. Каждый слой является бесконечно широким, описывается толщиной и рядом оптических характеристик.

Результатами моделирования являются значения следующих физических величин:

- пространственное распределение интенсивности рассеянного назад излучения на поверхности среды;
- пространственное распределение интенсивности рассеянного вперед излучения на задней границе среды;
- объемное распределение поглощенной интенсивности в среде.

Практическое решение поставленной задачи с использованием современных процессоров требует значительного времени вычислений. В связи с чем, актуальной является задача ускорения этого процесса.

Общее описание алгоритма решения задачи

Одним из подходов к решению задачи распространения света в многослойных средах является метод статистического моделирования Монте-Карло [1, 4]. Под методом Монте-Карло понимается совокупность приемов, позволяющих получать необходимые решения при помощи многократных случайных испытаний. Оценки искомой величины выводятся статистическим путем.

Применительно к задаче распространения излучения в многослойной среде метод Монте-Карло заключается в многократном повторении расчета траектории движения фотона в среде, исходя из задаваемых параметров среды.

Общий алгоритм решения задачи представлен на схеме [1] (рис. 1).

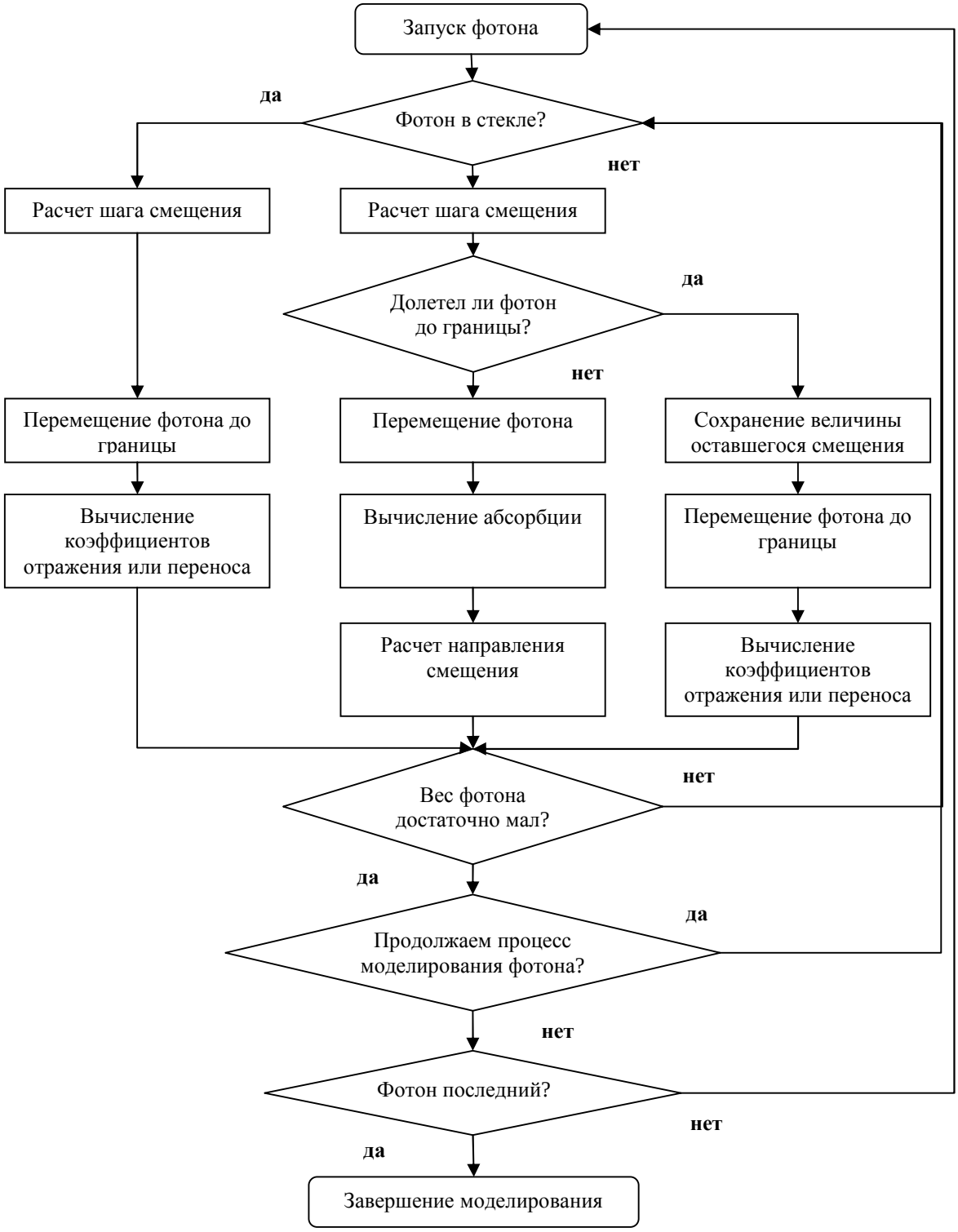

Рис. 1. Алгоритм моделирования распространения света в многослойной среде методом Монте-Карло

Для каждого фотона вышеуказанные действия выполняются независимо. Число фотонов в реальных экспериментах может быть достаточно большим (10^8 – 10^{10}), из чего можно предположить, что данная задача может быть эффективно реализована при использовании архитектуры графических адаптеров (GPU). В работе описывается

реализация представленного выше алгоритма на графическом адаптере с использованием технологии CUDA [2].

Схема распараллеливания и оптимизация

В рассматриваемой задаче необходимые для моделирования действия производятся независимо над каждым фотоном, что позволяет эффективно применить схему распараллеливания, при которой каждый поток вычисляет траекторию движения своего фотона. Поскольку число фотонов велико, предложенная схема позволяет загрузить все доступные вычислительные ядра графического адаптера.

Следует отметить, что для повышения производительности каждому потоку имеет смысл проводить вычисления сразу над группой фотонов (в текущей реализации размер группы фотонов равен 1000).

Для обеспечения сходимости метода моделирования распространения света необходимо, во-первых, наличие длинной последовательности различных случайных чисел, во-вторых, организация работы каждого конкретного вычислительного потока со своей подпоследовательностью, причем все эти подпоследовательности не должны пересекаться.

Генератор случайных чисел, реализованный в стандартной библиотеке языка программирования Си (функция rand) порождает недостаточно длинную последовательность случайных чисел, и к тому же не может быть использован при программировании на GPU. Поэтому было принято решение адаптировать алгоритм генератора MCG59 для работы в несколько потоков на графическом адаптере. В текущей реализации все потоки работают с одной последовательностью случайных чисел, но при этом каждый поток использует свои ($id + i * N$) элементы этой последовательности (здесь id – номер потока, N – общее число потоков).

Важным условием эффективности программ на GPU является правильное использование имеющихся типов памяти. В частности для решения данной задачи необходимы следующие наборы данных: информация о среде (хранится в разделяемой памяти), текущие характеристики фотона (хранятся в регистрах) и результирующие данные (имеют большой объем, хранятся в глобальной памяти). 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. Глобальная память, напротив, позволяет хранить большие объемы данных, но является относительно медленной. Однако в данной задаче обращений к результирующим данным немного, поэтому существенного влияния на производительность использование медленной памяти не оказывает.

Данные, относящиеся к результатам, являются общими для всех фотонов, откуда возникает необходимость обращения к этим данным одновременно из разных потоков, а значит, нужна синхронизация. В текущей реализации для этого используются атомарные операции, работающие с 64-битными целыми числами (для их применения версия compute capability графического адаптера должна быть 1.2 и выше).

Использование механизмов синхронизации обычно сильно уменьшает производительность приложений. Рассматриваемая задача не является исключением, однако здесь все зависит от размерности результирующих массивов, которая является входным параметром алгоритма. При малой размерности (100 элементов в каждом массиве) одновременных обращений к одним и тем же данным много, скорость работы приложения низкая. Но если увеличить число элементов каждого массива до 10 000 (тем самым повысив разрешающую способность задачи), то влияние синхронизации на производительность программы будет минимально.

Дополнительно следует отметить, что для решения требуемой задачи достаточно одинарной точности. А использование операций с одинарной точностью на GPU так же существенно ускоряет приложение. К тому же в данном случае возможно использование более быстрых, но менее точных математических функций, предоставляемых встроенной библиотекой CUDA.

Использование нескольких GPU

Технология CUDA позволяет использовать одновременно несколько графических адаптеров, установленных в рамках одного вычислительного узла. Каждый GPU имеет свой уникальный идентификатор, с помощью которого можно запускать выполнение задачи на выбранном устройстве. При этом для одновременной работы нескольких GPU необходимо создать на CPU определенное число параллельных потоков, чтобы каждый поток работал с собственным графическим адаптером. Способ создания потоков на центральном процессоре может быть любым, в частности в данной работе использовалась технология OpenMP.

В целях организации параллельной работы нескольких GPU, установленных на разных вычислительных узлах (например, в рамках вычислительного кластера), приведенный выше подход может быть обобщен. Для этого необходимо дополнительно применить одну из технологий параллельного программирования для систем с распределенной памятью, например, MPI [3].

Для использования нескольких графических адаптеров нам потребовалось организовать корректную работу с имеющимися данными:

- перед началом выполнения алгоритма создаются управляющие структуры, которые хранят данные для работы генератора случайных чисел (начальное смещение в генерируемой последовательности для конкретного устройства и общие данные генератора);
- перед началом выполнения алгоритма делается несколько копий структуры с информацией о моделируемой среде (входные данные задачи), каждая копия используется на своем GPU;
- структура состояния фотона создается и используется только в рамках вычислительного потока на графическом адаптере, поэтому не требует выполнения дополнительных действий при переходе на несколько GPU;
- промежуточные результаты накапливаются в памяти каждого из используемых графических адаптеров, а после окончания основного алгоритма суммируются на центральном процессоре.

Таким образом, рассматриваемый нами алгоритм позволяет создать работающую на нескольких GPU программу за небольшое количество достаточно простых шагов.

Результаты

В табл. 1 приведено время работы программы (в секундах) в зависимости от числа фотонов и вычислительного устройства.

Процессор: Intel Xeon E5520 2.27 ГГц, 4 ядра.

Количество ядер в процессоре: 4.

Количество процессоров: 2.

Объем оперативной памяти: 12 Гб.

Графический адаптер: Nvidia Tesla C1060.

Таблица 1. Время работы алгоритма (в секундах) в зависимости от числа фотонов и вычислительного устройства

Вычислительное устройство	Число фотонов				
	10^4	10^5	10^6	10^7	10^8
Xeon E5520, 1 ядро (1 core)	0,50	4,90	49,3	494	4951
2 x Xeon E5520, 8 ядер (8 cores)	0,11	0,94	7,60	74	730
Tesla C1060 (1 GPU)	0,06	0,11	0,60	5,13	50,7

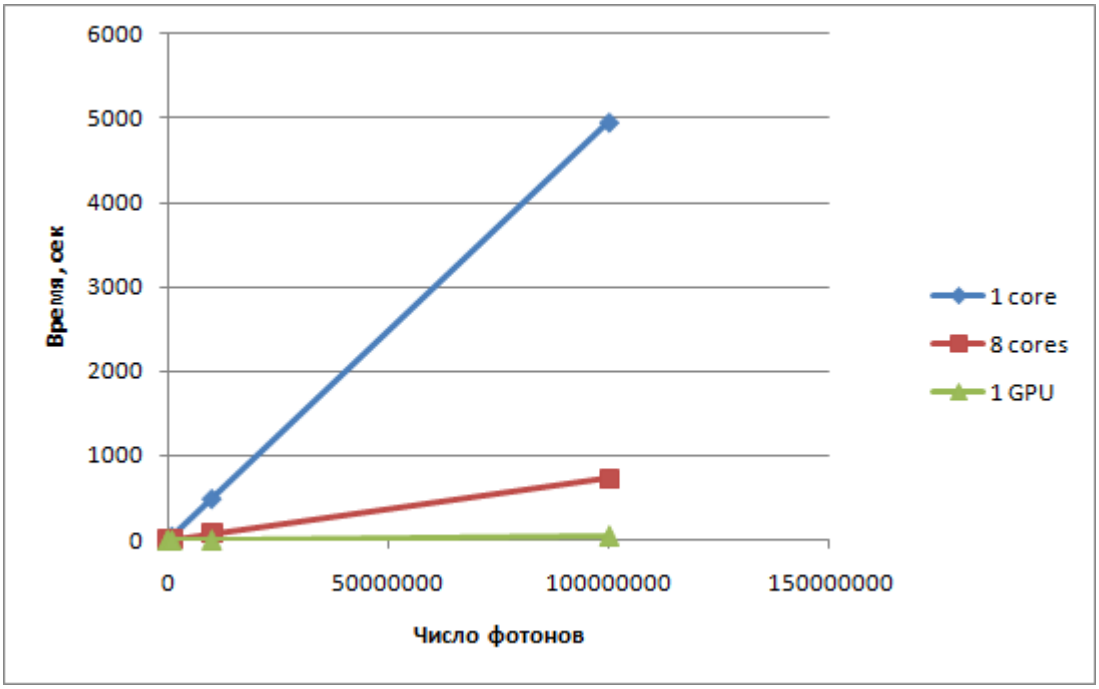

Рис. 2. Время работы алгоритма в зависимости от числа фотонов и вычислительного устройства

Таблица 2. Время работы алгоритма (в секундах) в зависимости от количества используемых GPU, число фотонов равно 10^8

Вычислительное устройство	Число используемых GPU			
	1	2	3	4
Tesla C1060	50.73	25.76	18.14	12.84

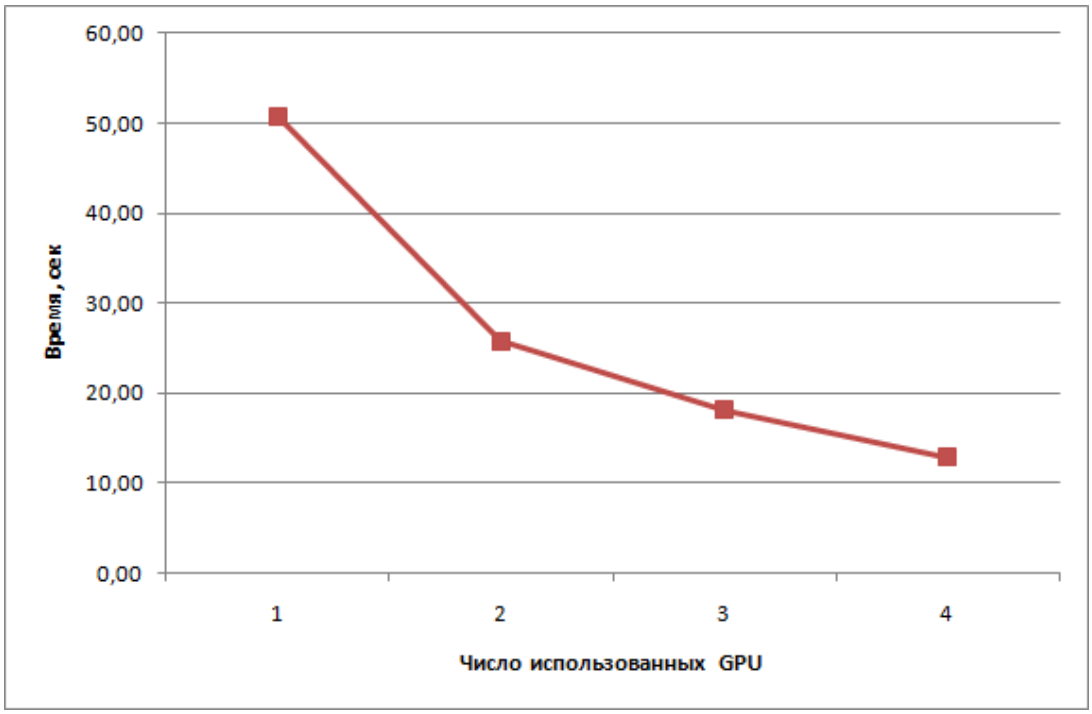

Рис. 3. Время работы алгоритма в зависимости от числа использованных GPU, число фотонов равно 10^8

Результаты проведенных экспериментов (табл. 1 и рис. 2) показывают, что использование графического адаптера и технологии CUDA существенно повышает производительность приложения, а потому является обоснованным.

А применение нескольких GPU для обработки данных (табл. 2 и рис. 3) позволяет дополнительно уменьшить время, необходимое для исполнения программы. Причем проведенные эксперименты говорят о хорошей масштабируемости задачи при одновременной работе на нескольких графических адаптерах.

Усовершенствование разработанного программного кода для расчета распространения излучения в средах со сложной геометрией позволит в перспективе эффективно использовать его как для моделирования работы различных систем оптической биомедицинской диагностики, так и для планирования лучевой терапии.

Авторы выражают благодарность Меерову Иосифу Борисовичу и Донченко Роману (Нижегородский госуниверситет им. Н.И. Лобачевского). Работа выполнена при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России», проект № 02.740.11.0839.

Литература

1. LihongWang, Steven L. Jacques. Monte Carlo Modeling of Light Transport in Multi-layered Tissues in Standard C, 1992.

2. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. Издательство "ДМК Пресс", 2010.

3. Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-Университет Информационных Технологий, Бином, 2007.

4. Соболь И.М. Численные методы Монте-Карло. Издательство "Наука", 1973.