Реализация алгоритмов цифровой фильтрации и масштабирования видеоизображений на специализированном векторном процессоре G80

Г. Н. Мельников
Томский государственный университет, Томск

Пришло время новой эры расчетов с помощью графических процессоров. Не секрет, что попытки переложить часть вычислений на GPU (Graphic Processor Unit) предпринимались и ранее, однако не имели значимых успехов. 16 февраля 2007 г. корпорация NVIDIA, которая является одним из мировых лидеров по производству графических процессоров, объявила о выпуске набора компонентов для разработчиков под названием CUDA (Compute Unified Device Architecture), а также компилятор языка C для расчета на собственных GPU. Это событие должно дать новый толчок для развития рынка аппаратных вычислений с использованием графических процессоров. По своей сложности GPU уже давно обогнали процессоры, хотя бы за счет использования большего количества транзисторов, а также своей изначальной ориентации.

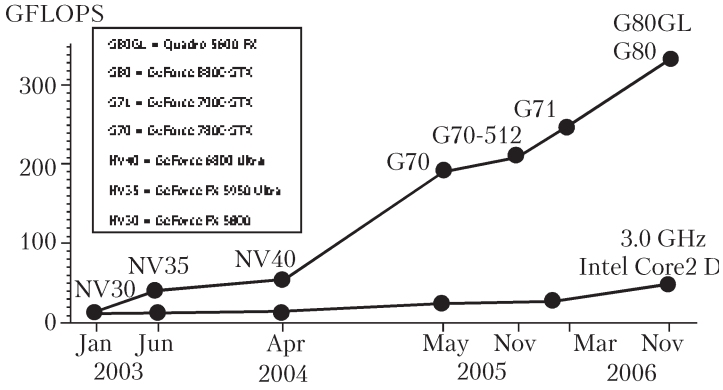


Рис. 1. Вычисление операций с плавающей точкой для CPU и GPU

Так что в современной системе видеокарта могла бы спокойно выступать в качестве помощника для решения тяжелых вычислительных задач. Технология CUDA призвана распараллелить вычислительные задачи между центральный процессором

(CPU) и GPU через стандартный язык программирования C. Пока что поддержка новой технологии присутствует в серии видеокарт GeForce 8800, однако она также будет реализована и в следующих решениях. Разработчики могут загрузить пока что тестовую версию CUDA с официального сайта NVIDIA [1].

Один из графических процессоров, поддерживающих CUDA-технологию, — это процессор G80 (он присутствует в видеокартах GeForce 8800 GTX (GTS)). Данный процессор предлагает огромные вычислительные ресурсы для обработки графической и неграфической информации. Его стоит рассматривать как дополнительное вычислительное устройство, с помощью которого можно разгрузить центральный процессор. Однако, не любую задачу выгодно реализовывать на GPU. GPU предназначен для решения тех задач, которые легко распараллеливаются. Для задач, которые выполняют одинаковые арифметические операции, но над разными данными.

Модель программирования

Программирование на процессоре G80 не стоит рассматривать так же, как и на векторных процессорах x86. Для эффективной работы, в отличие от x86, каждый из потоков, выполняющихся на G80, должен выполнять одни и те же операции, но над разными данными. Отсюда вытекает класс алгоритмов, хорошо ложащихся на данный процессор. Самым простым примером такого алгоритма является, например, независимая обработка каждого элемента массива. Кроме того, G80 специализирован под вычисления с плавающей точкой (тип float (32 бита) в языке C). Сложение, умножение, сложение плюс умножение в данном типе происходят всего за два такта.

Те части приложения, которые вычисляются много раз, но над различными данными, могут быть легко преобразованы в функцию, которая выполняется на устройстве посредством большого числа потоков (Threads). Такая функция в терминах CUDA называется кернелом (Kernel).

Потоки, которые выполняются в кернеле, организованы в таблицу блоков (Grid). Каждый блок содержит некоторое число потоков. Все потоки блока имеют доступ к одной кеш-памяти процессора, на котором они выполняются и могут синхронизироваться. Потоки разных блоков синхронизироваться не могут.

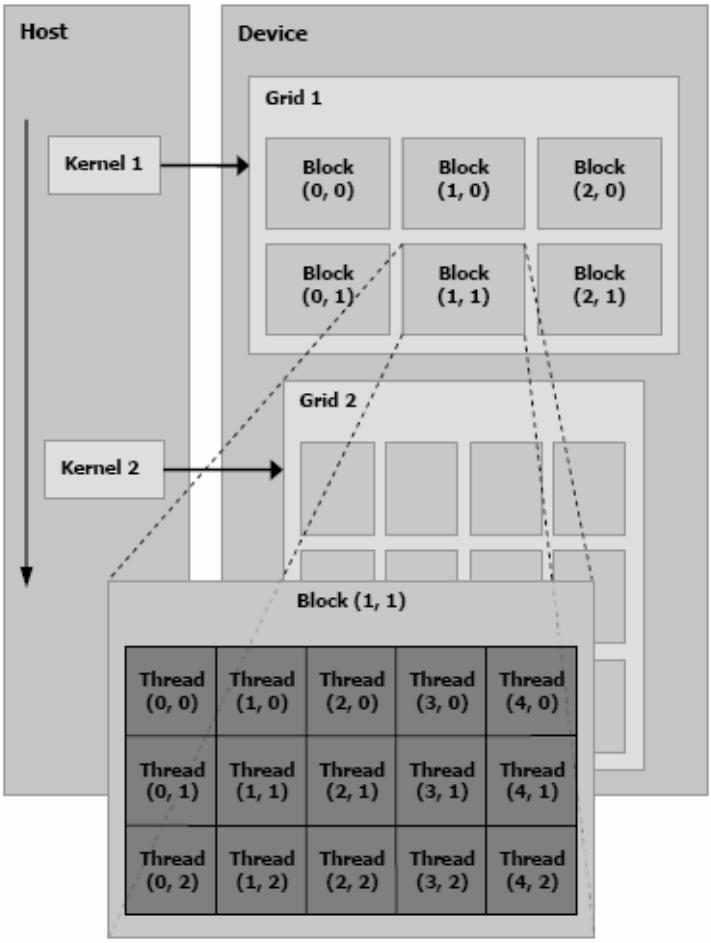


Рис 2. Организация системы потоков (threads) на G80

Блоки и потоки могут индексироваться одномерными, двумерными и трехмерными массивами (один из способов показан на рис. 1). Индексирование является чисто формальным, и размерность массива индексации стоит выбирать исходя из решаемой задачи. Каждый Kernel может предоставить номер блока и потока, который обрабатывается в текущий момент. Исходя из этого, можно однозначно определить часть данных, которую следует обработать.

Количество блоков и входящих в блок потоков должны быть выбраны так, чтобы максимизировать вычислительную мощь процессора. Это означает, что количество блоков должно быть не меньше, чем мультипроцессоров в устройстве. Кроме того, мультипроцессор будет простаивать (в моменты синхронизации и при чтении из памяти) при размещении в нем только одного блока. Поэтому количество блоков должно быть не менее чем в два раза больше, чем мультипроцессоров в устройстве. Кроме того, количество кэш-памяти, приходящейся на блок, должно быть не более половины от общего количества памяти, приходящейся на мультипроцессор. При полной загрузке кэш-памяти одним блоком не будет места в памяти для обработки иного.

Кроме того, нежелательно, чтобы потоки хаотично обращались к кэш-памяти процессора: их хаотичное обращение снизит вычислительную мощь процессора. Доступ к памяти должен быть выровненным [1].

Таким образом, общая схема программирования на G80 заключается в следующем:

1. Копирование данных в память устройства;
2. Копирование каждым потоком своих данных в кэш-память;
3. Обработка данных в кэше;
4. Копирование полученных данных в память устройства;

Модель памяти

G80 состоит из 16 (12) мультипроцессоров по 675 (600) МГц каждый. Каждый из процессоров имеет память следующего типа:

1. 32 битные регистры (8192 шт.)
2. Shared memory — кэш-память (16 Kb).
3. Константный кеш (только для чтения) (8 Kb).
4. Текстурный кеш (только для чтения) (8 Kb).
5. Глобальная память (DRAM), доступна для всех мультипроцессоров. Она не кэшируема.

Все потоки, которые выполняются на устройстве, имеют доступ только к DRAM памяти устройства.

Ниже проиллюстрирована модель доступа к памяти.

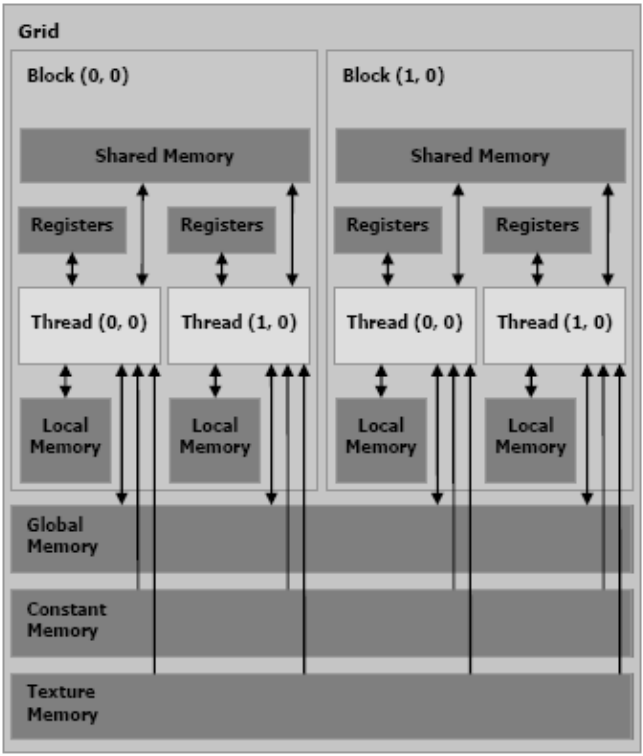


Рис. 3. Модель доступа к памяти

Модель вычисления

Таблица блоков вычисляется на устройстве посредством вычисления одного или нескольких блоков на каждом мультипроцессоре. Каждый блок разбивается на группы потоков, называемые варпами (warp). Каждый варп состоит из некоторого числа потоков (для G80 в варпе 32 потока). Планировщик потока периодически переходит от одного варпа к другому для максимизации использования вычислительных инструкций процессора. Таким образом, структура программы должна быть так разбита на потоки, чтобы каждые 32 потока выполняли одни и те же операции, но над разными данными (так как они выполняются одновременно). Следующие 32 потока могут производить иные вычисления.

Каждый мультипроцессор может обрабатывать несколько блоков параллельно. Это зависит от количества регистров и кэш-памяти, приходящейся на блок.

Заключение

Вычислительная мощь процессора G80 поражает: алгоритмы, которые можно эффективно на нем реализовать, могут выполняться в десятки раз быстрее, чем на центральном процессоре. К таким алгоритмам можно отнести алгоритмы, которые выполняют одни и те же операции, но над различными данными. Причем желательно чтобы количество памяти, необходимой для текущего вычисления, в одном потоке, было не велико (так как у процессора маленький кэш).

Одни из задач, которые можно эффективно решать с использованием CUDA технологии, это задачи, основанные на фильтрах и вейвлетах (например, задача подавления шума и масштабирования видеоизображений). Вейвлет-преобразование, требующее вычислений с плавающей точкой на GPU, будет выполняться в десятки раз быстрее, чем на центральном процессоре. Самая простая реализация данных алгоритмов с использованием технологии CUDA, выглядит следующим образом: мы создаем количество потоков, равных количеству пикселей в изображении, причем каждый поток обрабатывает соответствующий ему пиксель. Реализованный фильтр подавления шума (по времени и пространству), выполняется на устройстве в 50 раз быстрее, чем на центральном процессоре с полной оптимизацией кода. Однако, алгоритмы, быстро выполняющиеся на центральном процессоре, не стоит реализовывать на устройстве вследствие низкой пропускной способности памяти (с компьютера на устройство и обратно). Необходимо реализовывать такие алгоритмы, вычислительная мощь которых сможет компенсировать потери, связанные с пропусканной способностью памяти. Пиковая пропускная способность согласно техническим данным составляет 2 Gb/s, а реальная порядка 1—1.5 Gb/s.

Литература

1. [Электронный ресурс]. — Режим доступа : http://developer.nvidia.com/object/cuda.html, свободный.