Лекция 10: Графические процессоры (ГП)

1 Архитектура

Большая часть логических элементов центральных процессоров (ЦП) отведена для кеширования памяти и контроллера. Это позволяет ядрам ЦП быстро выполнять сложные вычислительные процедуры, имея быстрый доступ к данным в памяти. На графических процессорах (ГП), большее количество логических элементов отведено для вычислений, а модули контроля процедур и кеша уменьшены (Рис. 1). Благодаря этому, плотность Арифметических Логических Устройств (АЛУ) на ГП существенно выше, чем на ГП. Для того, чтобы загрузить все АЛУ вычислениями, ГП должен выполнять много операций одновременно. Это осуществляется при помощи вычислительных потоков, каждый из которых выполняет те же самые арифметические операции, но на разных элементах массива данных. Например, на ЦП сложение двух векторов размерности M необходимо осуществлять в цикле. При этом все элементы складываются последовательно один за другим. На ГП, подобная процедура может быть выполнена в M независимых потоках, каждый из который получает один элемент результирующего вектора. Таким образом, все элементы суммы могут быть получены одновременно и время подобной операции соответствует времени выполнения одного сложения против M сложений на ЦП. Все АЛУ на ГП сгруппированы в мультипроцессоры. На самых новых ГП от компании NVidia, каждый мультипроцессор состоит из 32 АЛУ, собственного кеша и контроллера. Например, ГП GeForce GTX 580 имеет 16 мультипроцессоров, а общее число АЛУ на этом устройстве - 512. Сам ГП расположен на отдельной печатной плате, которая также обладает и памятью DRAM (Dynamic

Random Access Memory). Размер этой памяти зависит от конкретного устройства, и обычно составляет $1 - 1,5$Гб для обычных устройств (карт GeForce, используемых главным образом для обработки трёхмерных изображений), но может достигать и 6Гб на устройствах, специализированых под вычисления (карты Tesla). Пропускная способность памяти графических карт ($\sim$ 200Гб/сек) используется всеми вычислительными потоками одновременно, и, т.к. размер кеш-памяти на мультипроцессоре (64Кб) существенно уступает размеру кеш-памяти на ЦП ($\sim$ 2Мб), обращения к памяти должны быть сгруппированы для достижения максимальной производительности. Если доступ к данным не будет сгруппирован между потоками, ГП будет простаивать в ожидании новой порции данных из-за задержек в доступе к глобальной памяти устройства. С другой стороны, одновременно на ГП могут сосуществовать до 800 000 потоков, часть из которых может выполнять арифметические операции, пока другие находятся в ожидании новой порции данных. Это позволяет эффективно использовать вычислительные возможности ГП, пиковая производительность которых может достигать 1ТФлопа. Для того, чтобы достигнуть подобной производительности, численный алгоритм должен быть хорошо параллелизуемым и вычислительно ёмким.

В биомолекулярном моделировании с использованием молекулярной динамики или динамики Ланжевена, взаимодействия между частицами описываются при помощи потенциальной функции (силового поля). Функциональная часть силового поля одинакова для всех частиц в системе, а уравнения движения всех частиц могут быть численно проинтегрированы независимо друг от друга. Благодаря такому соответствию между ОКМД (Одиночный поток Команд, Множественный поток Данных; англ. Single Instruction Multiple Data, SIMD) архитектурой ГП и вычислительными процедурами молекулярной динамики, численные алгоритмы последней могут быть успешно реализованы на ГП.

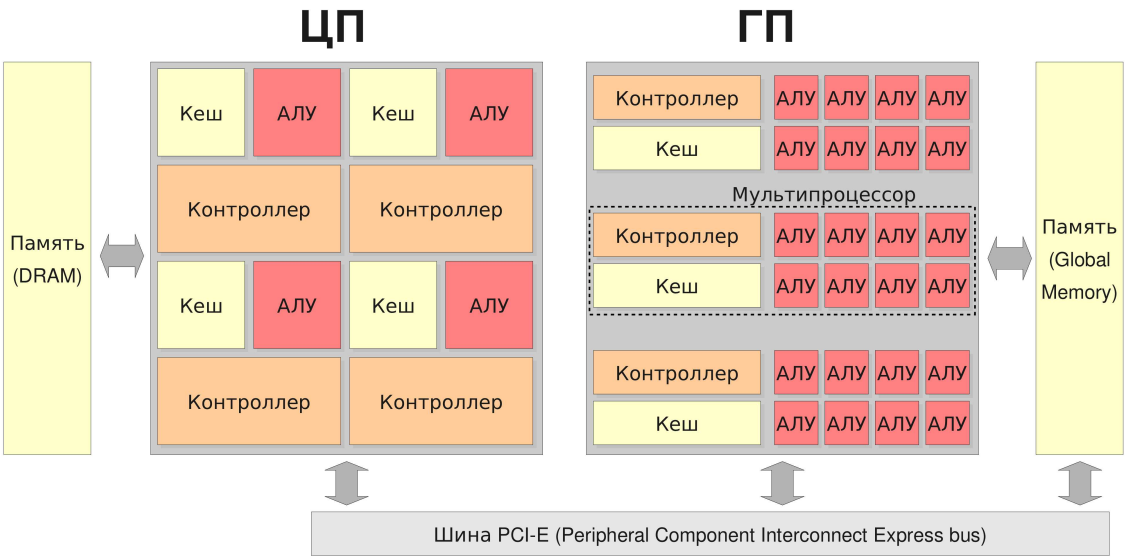

Рис. 1: Архитектура центрального (слева) и графического (справа) процессоров. На ЦП, значительная часть поверхности микро-чипа занята кеш-пямятью и контроллером. ЦП также оснащён большим количеством памяти DRAM (Dynamic Random Access Memory). Арифметические Логические Устройства (АЛУ) на ГП, сгруппированные в мультипроцессоры, каждый из который обладает своими контроллером и кеш-памятью. Это позволяет существенно увеличить плотность вычислительных единиц на микро-чипе. ГП обладает отдельной памятью DRAM, которую обычно называют глобальной памятью устройства. Взаимодействие между ЦП и ГП осуществляется через шину PCI Express.

2 Программная модель CUDA

Многие научные приложения оперируют с большим количеством данных, а выполнение такой программы требует пропорциональных вычислительных затрат, так как операции должны проводиться на всех элементах входного массива данных. Например, обработка изображения, где одна и та же функция применяется к разным фрагментам рисунка или различные физические задачи, где рассчитываются взаимодействия между большим количеством элементов системы. В таких задачах, расчёт производится по математическим формулам, применяемым ко всем элементам массива данных (массива точек, составляющих изображение или набора координат элементов системы). При этом, вычисление одного из элементов этого массива может выполнятся независимо от других. В случаях, когда

арифметические операции могут выполняться на всех элементах массива независимо, говорят, что задача обладает хорошим параллелизмом данных.

В качестве простого примера можно привести операцию сложения двух векторов большой размерности N. Расчёт суммарного вектора в данном случае может быть посчитан в N независимых сложений - по одному для каждого элемента i. При этом, два элемента i и j ($i \neq j$) могут быть рассчитаны как последовательно, так и одновременно. Очевидно, что последовательный расчёт займёт время, необходимое для проведения N операций сложения, в то время как одновременный расчёт на N вычислительных элементах будет занимать время, необходимое для проведения 1-ой операции сложения. Теоретически, параллельное сложение в данном случае может быть в N раз быстрее, если аппаратное обеспечение содержит достаточное количество Арифметических Логических Единиц (АЛУ).

В программировании на CUDA вычислительная система состоит из центрального процессора (ЦП) и устройства (чаще всего ГП). Часть программы не обладающая достаточным параллелизмом данных выполняется на центральном процессоре, а обладающая - на графическом. Программа, написанная на CUDA содержит обе части, которые разделяются в процессе компиляции. Часть программного кода, предназначенная для выполнения на ЦП, отделяется и передаётся обычному компилятору (например, gcc), а предназначенная для ГП - специализированному компилятору (nvcc). Вся программа представляет собой код, написанный на языке программирования C, расширенного специальными указателями на данные, хранимые на ГП, и функциями, называемыми ядрами (англ. kernel). Код, выполняемый на ЦП, может копировать данные на ГП, используя соответствующие указатели, и вызывать ядра для выполнения операций над этими данными.

Ядра обычно выполняются в большом количестве потоков для того, чтобы использовать высокий параллелизм данных. В примере со сложением векторов, всё сложение может быть выполнено при помощи одного вычислительного ядра, а количество потоков будет равно размерности вектора N. Аппаратная часть ГП построена таким образом, что потоки на ГП могут эффективно создаваться и завершаться. Это является важным отличием подхода к программированию на ГП, так как время создания N параллельных потоков на ЦП

было бы слишком существенным, чтобы получить выигрыш в производительности в данной задаче, даже при наличии достаточно большого количества вычислительных единиц.

Простейшая программа, написанная на CUDA, выглядит следующим образом. Выполнение начинается на ЦП, который выделяет память на ГП и копирует в неё исходные данные. Затем, вызывается ядро, которое обрабатывает эти данные согласно реализованным в нём процедурам. При этом, количество потоков, используемое при выполнении ядра, задаётся изначально. После чего выполнение программы возвращается на ЦП, который копирует результат вычислений из памяти ГП. В более сложных реализациях несколько ядер могут запускаться последовательно, ЦП может прерывать выполнение программы, если получен достигнутый результат, рекурсивно вызывать, чередовать ядра и так далее.

В CUDA, центральный и графический процессоры обладают отдельными областями данных. Это соответствует устройству компьютера (аппаратной части), в котором ГП располагается на отдельной печатной плате. Для того, чтобы обеспечить ГП данными, часть программы, работающая на ЦП, выделяет память как в своей области, так и в области ГП. После этого в распоряжении программиста оказываются два указателя на две области памяти: на память ЦП, с которой он может работать как обычно, и на память ГП, обращение к которой напрямую вызовет ошибку. Как только область памяти на ЦП заполнена исходными данными, они могут быть скопированы по указателю в памяти ГП при помощи специальных процедур CUDA. На аппаратном уровне такое копирование данных осуществляется через шину PCI-Express (Рис. 1).

На ГП, доступ к данным тесно связан с архитектурой устройства. Во-первых, ГП состоит из мультипроцессоров, каждый из которых имеет свою собственную кеш-память. На графических картах Fermi каждый мультипроцессор оснащён 32 вычислительными ядрами (АЛУ), общее количество которых может достигать 512 на самых мощных видеокартах (GeForce GTX 580). В CUDA это отражается на иерархии потоков выполнения, которые сгруппированы в блоки. Количество потоков в блоке (размер блока) может достигать 1024, превышая количество АЛУ на мультипроцессоре. Это позволяет ГП эффективно загружать вычислительные ресурсы: пока некоторые потоки ждут данных, другие могут быть

заняты расчётами. Количество блоков на ГП может достигать 65535. Ограниченные внутри одного блока потоки выполняются на одном и том же мультипроцессоре и имеют доступ к его кеш-памяти, которая может быть использована либо для хранения постоянных (постоянная память), либо для обмена данными внутри одного блока потоков (разделяемая память), либо для хранения данных конкретного потока (локальная память). Постоянная память доступна всем потокам блока только на чтение, разделяемая память - на чтение и запись, локальная память используется потоками для хранения промежуточных значений. Выполнение потоков внутри блока могут также быть синхронизировано. Общая память ГП (глобальная память) расположена на печатной плате вне микрочипа и является аналогом DRAM центрального процессора. Хотя эта память открыта для доступа всем потокам, выполняющимся на всех мультипроцесоорах, доступ к ней существенно медленнее. Поэтому эффективное использование кеша мультипроцессоров очень важно для достижения хорошей производительности и является одной из главных стратегий её повышения. CUDA позволяет разработчику контролировать потоки данных и явно задавать область памяти для хранения тех или иных данных. Адресация данных в CUDA осуществляется при помощи номеров (индексов) потока и блока. В примере со сложением векторов, каждый поток будет работать с элементом $i = i_b \times B + i_t$, где i_b - номер блока, B - его размер, а i_t - номер потока. В более сложных примерах, сетка потоков может быть двух и даже трёхмерной. Это удобно, например, при расчёте суммы или произведения матриц.

Таким образом, на ГП есть потоки выполнения, которые аппаратно распределяются по АЛУ устройства. Кроме регистров памяти, все потоки могут использовать кеш-память мультипроцессора для хранения локальных данных. Потоки сгруппированы в блоки, которые распределяются по мультипроцессорам. Каждый блок может использовать кеш-память мультипроцессора для синхронизации и обмена данных между своими потоками. Все потоки, выполняющиеся на ГП, могут использовать глобальную DRAMM-память устройства.

3 Подходы к оптимизации

Иерархическая структура данных на ГП открывает широкие возможности по оптимизации программного кода. Во-первых, количество регистров на мультипроцессоре ограничено, поэтому, чтобы избежать использования более медленной локальной памяти, программист должен контролировать количество используемых локальных величин внутри ядра. Во-вторых, разделённая память может быть использована для явного кеширования данных. В-третьих, количество обращений к глобальной памяти должно быть минимизировано, либо распределено по программному коду так, чтобы все потоки могли выполнять арифметические операции в процессе ожидания данных. Также, в силу наличия аппаратного кеша, обращение к глобальной памяти должно быть локализовано - соседние потоки должны обращаться к соседним ячейкам памяти. Большую роль играет также и упорядочевание обращений к глобальной памяти, которое позволяет ГП объединять запросы от нескольких потоков в один. Существенным фактором, который может ограничить производительность программы, является скорость передачи данных от ЦП к ГП, которая ограничена шиной PCI-Express. Программист должен, по возможности, избегать копирования данных между ЦП и ГП. Этого можно добиться переносом всего программного кода на ГП, используя ЦП лишь для управления потоком вычислений и вывода результатов.

В силу того, что выполнение одного блока потоков целиком занимает мультипроцессор, программист должен по возможности избегать ветвлений. Это также важно и из-за аппаратных особенностей ГП, на котором группы по 32 потока должны выполнять идентичные процедуры. Если в группе возникнет ветвление хотябы в одном потоке, остальные будут простаивать.

Программный код ядра выполняется в множестве потоков. Поэтому оптимизация ядер - очень важный шаг к достижению максимально возможной производительности. Для оптимизации можно использовать как общепринятые методики, так и специфические для ГП. Например, на ГП существуют аппаратно-ускоренные специальные функции, позволяющие существенно ускорить расчёт тригонометрических операций, взятие экспоненты, деление. Эти функции должны быть использованы с осторожностью, так как они обладают несколь-

ко меньшей точностью. Также, программист может запросить данные из глобальной памяти задолго до того, как они понадобятся для выполнения программы. Это позволит ГП заниматься расчётами в процессе ожидания новых данных.