УДК 681.3

С.Д. ПОГОРЕЛЫЙ, Ю.В. БОЙКО, М.И. ТРИБРАТ, Д.Б. ГРЯЗНОВ

АНАЛИЗ МЕТОДОВ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ КОМПЬЮТЕРОВ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ И ПРОГРАММНО-АППАРАТНОЙ ПЛАТФОРМЫ CUDA

Abstract. *Analysis methods for improving the performance of computers made using graphics processors and software and hardware platform CUDA. Ability to use graphics processors is considered to perform mathematical calculations. Features of the calculation analyzed using adapters.*
Key words: *CUDA, GPGPU, the video adapter, the video processor, the central processor, calculations with use of the graphic processor.*

Анотація. *Виконано аналіз методів підвищення продуктивності комп'ютерів з використанням графічних процесорів і програмно-апаратної платформи CUDA. Розглянуто можливість використання графічних процесорів для виконання математичних розрахунків. Проаналізовано особливості виконання розрахунків з використанням відеоадаптерів.*
Ключові слова: *CUDA, GPGPU, відеоадаптер, відеопроцесор, центральний процесор, розрахунки з використанням графічного процесора.*

Аннотация. *Выполнен анализ методов повышения производительности компьютеров с использованием графических процессоров и программно-аппаратной платформы CUDA. Рассмотрена возможность использования графических процессоров для выполнения математических расчетов. Проанализированы особенности выполнения расчетов с использованием видеоадаптеров.*
Ключевые слова: *CUDA, GPGPU, видеоадаптер, видеопроцессор, центральный процессор, расчеты с использованием графического процессора.*

1. Вступление

С появлением в последние годы значительного количества видеоадаптеров компании NVIDIA с поддержкой новой программно-аппаратной платформы CUDA (Compute Unified Device Architecture) появилась возможность использования этой платформы для широкого спектра вычислительных задач. Если задача допускает разделение на множественные потоки, то более производительным будет выполнить ее на графическом процессоре (Graphics Processing Unit (GPU)), чем на центральном (Central Processing Unit (CPU)). Целью данной статьи является анализ методов повышения производительности компьютеров с использованием графических адаптеров и новой программно-аппаратной платформы CUDA.

Эмпирический закон Мура в исходной формулировке выполнялся с 1965 года до недавнего времени. Его формулировка: «каждые два года число транзисторов, наиболее эффективно с точки зрения себестоимости размещаемых на единице площади микросхемы, будет удваиваться» [1]. Начиная с XXI столетия, такая формулировка, фактически означающая экспоненциальный рост мощности вычислительных микросхем каждые 24 месяца, стала изменяться.

Увеличение частот и количества транзисторов, а, соответственно, и производительности центральных процессоров, последнее время замедлилось и практически остановилось (рис. 1). Из-за физических ограничений (технологического процесса, размерных эффектов) и высокого энергопотребления в связи с добавлением в процессор множества дополнительных транзисторных блоков дальнейший, «классический», способ наращивания мощности существенно затруднен. Число транзисторов еще возрастает, но уже на различных ядрах.

ISSN 1028-9763. Математичні машини і системи, 2010, № 1

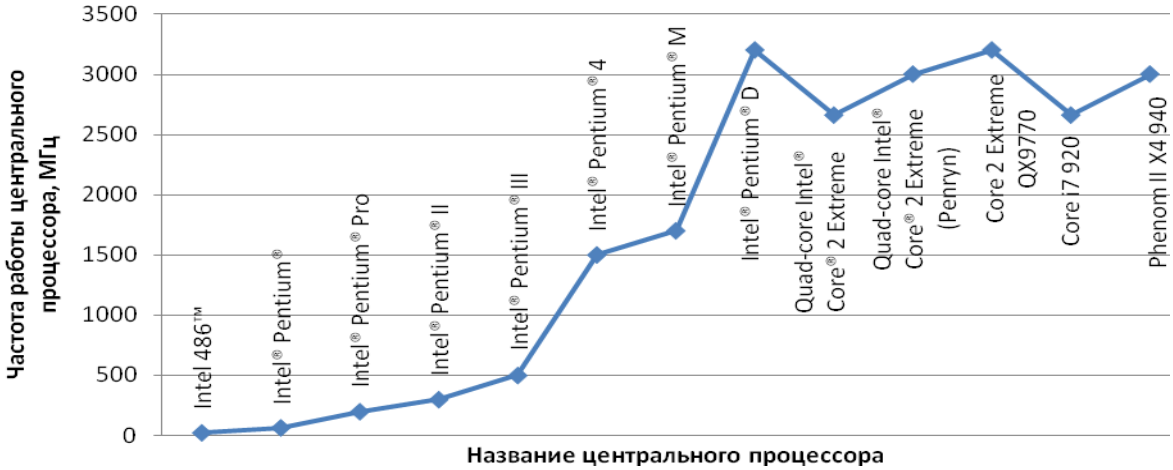

Рис. 1. Рост частоты в различных поколениях центральных процессоров

Процессоры с несколькими ядрами – это процессоры на кремниевой подложке либо процессоры, в одном корпусе которых располагаются не менее двух независимых вычислительных ядер [2, 3]. Добавление новых ядер и развитие многоядерных архитектур связано с тем, что наращивание тактовой частоты одноядерных процессоров приводит лишь к незначительному увеличению производительности, но при этом резко увеличивается энергопотребление и, соответственно, резко увеличивается тепловыделение (практически вся мощность процессора преобразуется в тепло). Таким образом, продолжение классического либо экстенсивного пути развития процессоров привело бы к тому, что температура их поверхности приблизилась бы к температуре Солнца [4].

При решении проблемы охлаждения остается критичным соотношение цена/производительность, что вызвано ростом затрат на производство новых процессоров, уровень производительности которых практически не будет отличаться от производительности процессоров предыдущего поколения.

2. Новые подходы к проблеме увеличения производительности

Видеоадаптеры в современном понимании уже не просто устройства или блоки персонального компьютера, которые отвечают за вывод информации на экран монитора. Это платы, которые имеют свой процессор, свою память, а к некоторым графическим адаптерам требуется подключение дополнительного питания: наиболее мощные потребляют до 300Вт [5].

Графические процессоры работают на более низких тактовых частотах, чем центральные процессоры, а некоторые исполнительные блоки в GPU – на частотах до 1,5 ГГц, что сопоставимо с частотой работы некоторых CPU [6]. При этом графические процессоры по количеству транзисторов на кристалле не уступают центральным процессорам (табл. 1).

Память видеоадаптера имеет меньшие максимальные объемы (до 4 Гб), чем оперативная память, но быстрее ее. Современные типы оперативной памяти и их пропускные способности представлены в табл. 2.

Таблица 1. Характеристики современных видеоадаптеров

Видеоадаптер	Radeon HD 4870	Radeon HD 4870 X2	Radeon HD 5870	GeForce GTX 260	GeForce GTX 285	GeForce 8600 GT
Технологический процесс (нм)	55	55	40	65	55	80
Число транзисторов (млн)	956	2x956	2150	1 400	1400	289
Частота ядра (МГц)	750	750	850	576	648	540
Частота потоковых процессоров (МГц)	750	750	850	576	1476	1190
Число потоковых процессоров	800	2x800	1600	192	240	32
Частота памяти (МГц)	900 (3600)	900 (3600)	1200 (4800)	1000 (2000)	1100 (2200)	700 (1400)
Ширина шины памяти (бит)	256	256	256	448	512	128
Объем памяти (Мбайт)	512 GDDR5	2x1024 GDDR5	1024 GDDR5	896 GDDR3	1024 GDDR3	256 GDDR3
Пропускная способность (Гбайт/с)	115,0	2x115,0	153,6	112,0	140,8	22
Производительность (GFlops)	1200	2x1200	2720	715	1063	76

Таблица 2. Современные типы оперативной памяти

Тип оперативной памяти	DDR3-800	DDR3-1066	DDR3-1333	DDR3-1600	DDR3-1800	DDR3-2000	DDR3-2133	DDR3-2400
Частота шины (МГц)	400	533	667	800	900	1000	1066	1200
Название модуля	PC3-6400	PC3-8500	PC3-10600	PC3-12800	PC3-14400	PC3-16000	PC3-17000	PC3-19200
Пропускная способность (МБ/с)	6400	8533	10667	12800	14400	16000	17066	19200

Необходимость рационального использования вычислительных ресурсов видеоадаптеров требует создания механизма, с помощью которого можно использовать указанные возможности. Это привело к появлению первых технологий, позволяющих осуществлять неграфические расчёты общего назначения (General-Purpose computation on Graphics Processing Unit (GPGPU)). Одной из первых таких технологий были шейдеры.

Шейдер (Shader) – это программа для одной из ступеней графического конвейера (аппаратно-программный комплекс визуализации трёхмерной графики), используемая в трёхмерной графике для определения окончательных параметров объекта или изображения [7, 8]. С появлением первых таких программ стало возможным задавать обработку пикселей и вершин многогранников с помощью кода, написанного на специальном ассемблере. При этом обработка велась параллельно и независимо друг от друга. Все вычисления над вершинами и пикселями производились с помощью 32-битовых чисел с плавающей запятой.

Аппаратные и программные платформы для выполнения неграфических расчетов на видеоадаптерах развивались и развиваются очень динамично. Два основных производителя видеопроцессоров, NVIDIA и AMD, разработали и анонсировали платформы CUDA [9] и ATI Stream (первоначально известную как CTM (Close To Metal)) [10] соответственно. В отличие от предыдущих парадигм программирования GPU, эти программные платформы выполнены с учётом прямого доступа к аппаратным возможностям видеокарт. Платформы не совместимы между собой, CUDA – это расширение языка программирования C, а ATI Stream – виртуальная машина, исполняющая ассемблерный код. Однако обе платформы ликвидировали важные ограничений предыдущих моделей GPGPU [11].

DirectX – это набор функций интерфейса прикладного программирования (Application Programming Interface (API)), разработанных для решения задач, связанных с игровым и видеопрограммированием для программных платформ Microsoft Windows [12].

Открытая графическая библиотека (Open Graphics Library (OpenGL)), соответствующая спецификации, определяющей независимый от языка программирования кросс-платформенный программный интерфейс для написания приложений, использующих двухмерную и трёхмерную компьютерную графику [13].

Для решения на GPU реальной задачи с использованием интерфейсов трехмерной графики DirectX и OpenGL, а также шейдерных программ, обрабатываемые данные необходимо представить в виде двухмерных массивов – текстур. Их необходимо загрузить и скомпилировать саму программу-шейдер, установив ее параметры и выходные буферы памяти, что делается при помощи центрального процессора, и только после этого исполнить шейдер [14].

Открытый язык вычислений (Open Computing Language (OpenCL)) – программное обеспечение для написания программ с параллельными вычислениями на различных GPU и CPU. Программное обеспечение состоит из языка программирования Си (основывается на стандарте C99) и интерфейса прикладного программирования (API). OpenCL является полностью открытым стандартом. Преимущество кросс-платформенной поддержки исключает автоматические инструменты, например, управление памятью, которые присутствуют в CUDA [15].

Следует отметить, что GPU никогда не заменит CPU. Центральный процессор в персональном компьютере выполняет все расчеты. Это универсальный процессор, мощность которого в различных специфических задачах может существенно отличаться от узкоспециализированных для конкретных целей процессоров, таких как GPU. Однако теперь появилась возможность использования и этих ресурсов [16].

3. Анализ графических процессоров

Анализ современных подходов к проектированию архитектуры и созданию центральных процессоров говорит о том, что вектор развития действительно меняет свое направление от экстенсивных к новым подходам и тенденциям [17].

Дальнейший прирост производительности следует ожидать именно от симбиоза GPU и CPU при использовании каждого в специфических для него задачах. GPU – это сопроцессор для CPU.

В архитектуре CPU уже используются параллельные подходы к обработке для повышения производительности: конвейеры, эмуляция нескольких логических ядер на базе одного физического (Multithreading), потоковые SIMD [18] расширения процессоров (Streaming SIMD Extensions (SSE)) [19].

У видеопроцессоров вычислительные задачи однородны и распараллеливаются изначально. На входе GPU берет группу вершин многогранников, производит все необходимые расчеты и на выходе выдает группу пикселей (например, один кадр). Обработка вершин многогранников и пикселей независима и происходит одновременно, параллельно, отдельно друг от друга. Поэтому подобная параллельная организация работы в GPU требует большого количества исполнительных блоков. Экран монитора, к примеру, имеет разрешение 1024x768. Тогда видеоадаптер должен обработать 786432 пикселя, чтобы заполнить экран, но при частоте 30 кадров в секунду видеоадаптер обрабатывает 23 592 960 пикселей в секунду.

4. Аппаратная платформа графических процессоров G80и GT200

Организационная структура процессора NVIDIA G80 имеет следующий вид [20] (рис. 2). Аббревиатуры на этом рисунке и на следующих указаны ниже.

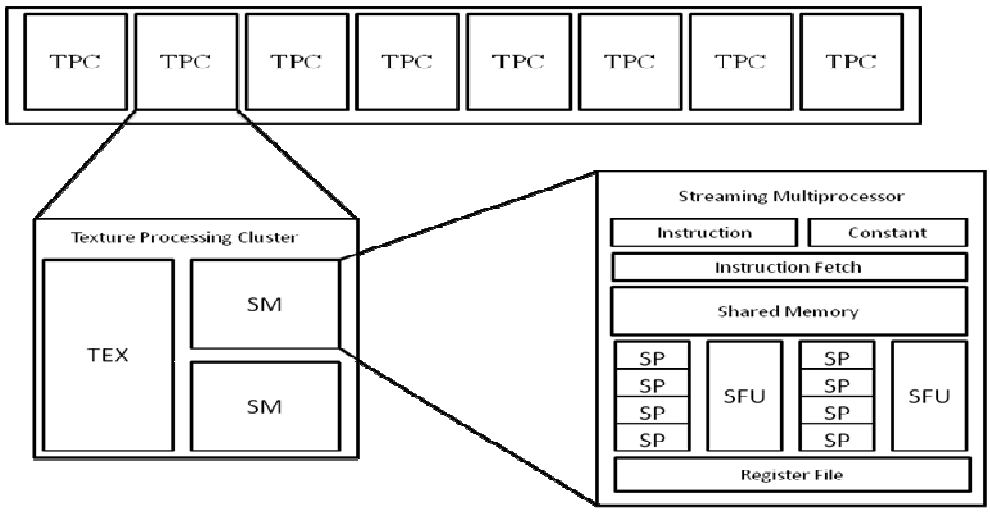

Рис. 2. Архитектура видеопроцессора NVIDIA G80

Архитектура основывается на массиве потоковых процессоров NVIDIA (Streaming Processor Array (SPA)), в который объединяются кластеры текстурных процессоров (Texture Processor Clusters (TPC)). В видеоадаптере 8800 GTX используется восемь кластеров, 8800 GTS – шесть и т.д. В каждом кластере есть текстурный блок (TEX) и два потоковых мультипроцессора (Streaming Multiprocessors (SM)) [14].

Каждый потоковый мультипроцессор состоит из таких элементов:

• кэш-памяти (Cache), инструкций и данных – констант;

• общей памяти (общая/разделяемая память (Shared memory)) – объем памяти изменяется в зависимости от видеопроцессора;

• потоковых процессоров (Streaming Processor (SP)), по сути – это скалярные арифметико-логические устройства (Arithmetic Logic Unit (ALU)). В каждом потоковом мультипроцессоре графического процессора G80 их восемь;

• специализированных функциональных блоков (Special Function Unit (SFU)) – модулей для вычисления математических функций sin, cos, exp и т.д.;

• регистров – все регистры 32-разрядные.

Для дальнейшего рассмотрения будем использовать введенный фирмой NVIDIA термин warp. Warp в CUDA представляет собой группу из 32 потоков (Thread) и является минимальным объёмом данных, обрабатываемых SIMD-способом в мультипроцессорах CUDA [21].

Два специализированных функциональных блока представляют собой модули, выполняющие инструкции по принципу SIMD – одна инструкция применяется ко всем потокам в warp. Фирма NVIDIA называет такой способ выполнения «одна инструкция, много потоков» (Single Instruction Multiple Threads (SIMT)).

Следует отметить, что потоковые процессоры работают на значительно большей частоте, чем само ядро. Например, в графическом видеоадаптере 8800 Ultra на частоте около 1,5 ГГц, а в GTX 280 – 1,3 ГГц.

Тактовые частоты графических процессоров ниже, чем у центральных процессоров, но благодаря большому количеству скалярных процессоров их производительность значительно превосходит производительность центральных процессоров [14]. Современные видеоадаптеры имеют пиковую производительность от 100 до 1000 GFlops (рис. 3), что при возможности установки в один системный блок компьютера четырех видеоадаптеров позволяет получить пиковую производительность 4,14 TFlops на одном компьютере [22].

Рис. 3. Теоретическая производительность GPU

Такая производительность достигается за счет аппаратных и программных платформ видеоадаптера, специализированного для параллельной обработки данных.

Соответствие аппаратных платформ и видеоадаптеров, в которых они используются, можно увидеть в табл. 3.

Таблица 3. Использование архитектур в видеоадаптерах NVIDIA

Название архитектуры	NV30	NV35	NV40	G70	G71	G80	G92	GT200
Название видеоадаптера	GeForce FX 5800	GeForce FX 5950 Ultra	GeForce 6800 Ultra	GeForce 7800 GTX	GeForce 7900 GTX	GeForce 8800 GTX	GeForce 9800 GTX	Tesla C1060

Потоковый мультипроцессор представляет собой SIMD-процессор разрядностью 32 бита. При векторной обработке каждая команда сопровождается множеством данных, над которыми эта

команда выполняется. Каждый потоковый мультипроцессор состоит из 8 потоковых процессоров. Таким образом, каждый графический процессор содержит 128 исполнительных блоков (8 кластеров текстурных процессоров, содержащих 2 потоковых мультипроцессора с 8 потоковыми процессорами (8x2x8)). То есть, если потоковые процессоры работают на частоте 1,512 ГГц (табл. 1), то можно рассчитать пиковую производительность одного видеоадаптера. Количество потоковых процессоров 128 необходимо умножить на частоту 1,512 ГГц и умножить на количество операций за такт – 3, которые выполняет графический процессор (128 x 1,512 ГГц x 3). Рассчитанная производительность составляет 581 GFlops (Floating point Operations Per Second (Flops) [23]).

При рассмотрении и анализе архитектуры графических процессоров G80 и GT200 можно заметить, что архитектура GT200 – это практически та же архитектура G80 с небольшими изменениями (рис. 4).

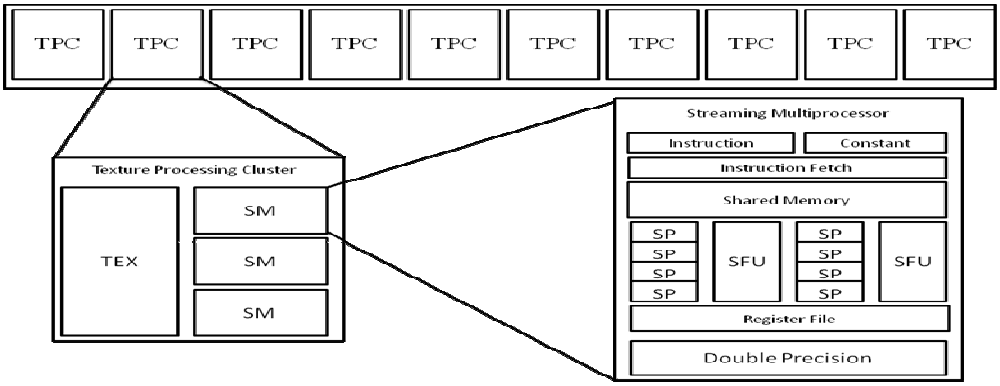

Рис. 4. Архитектура графического процессора GT200

У графического процессора GT200 фирмы NVIDIA увеличено число кластеров текстурных процессоров до 10, каждый по-прежнему оснащён текстурным блоком, но уже тремя потоковыми мультипроцессорами (3 модуля).

В каждый SM-модуль добавлен сопроцессор для вычислений с двойной точностью (Double Precision). Он может выполнять инструкции двойной точности, позволяющие перемножить два числа и прибавить к ним третье (Multiply+Add (MAD)) за такт. Указанным ранее способом расчет пиковой производительности модуля двойной точности дает результат 77,78 GFlops (1,296 ГГц x 10 кластеров x 3 потоковых процессора x 2 операции MAD). Таким образом, теоретически операции с двойной точностью в 12 раз выполняются медленнее, чем операции с одинарной точностью.

Последнее изменение, сделанное с потоковыми мультипроцессорами, касается поддержки двойной точности (64-битное число с плавающей запятой вместо 32-битного). Обычно дополнительная точность не используется в графических алгоритмах, но в некоторых научных приложениях двойная точность необходима.

Таким образом, производитель, добавляя или удаляя кластеры текстурных процессоров из графического процессора, может предложить необходимую производительность за определенную цену. Для параллельных вычислений и задач зависимость между количеством узлов процессора (ядер процессора, количества процессоров) и производительностью фактически будет линейна. Однако на производительность могут влиять и другие факторы, например, недостаточная полоса пропускания шины видеоадаптера PCI Express.

При сравнении использования физической площади кристалла центральным и графическим процессорами, видно, что ALU в GPU занимают практически всю площадь, в то время как в CPU – гораздо меньшую часть (рис. 5).

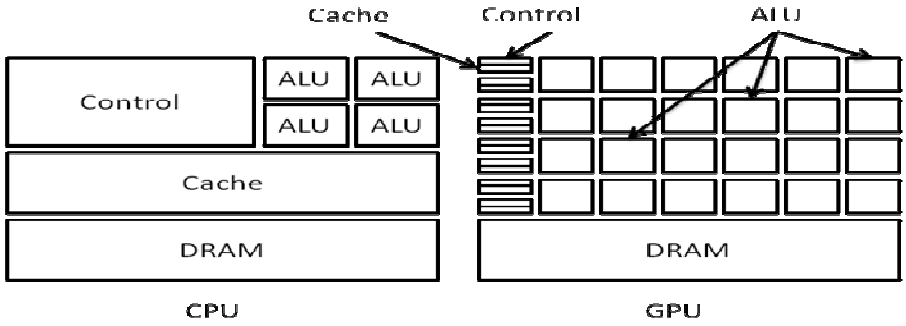

Рис. 5. Распределение площади физической поверхности на кристалле CPU и GPU между ALU, дополнительными блоками и кэш-памятью

В универсальных процессорах есть значительное число аппаратных блоков, которые необходимы для исполнения немногочисленных потоков команд. Соответственно транзисторы таких блоков занимают площадь кристалла так же, как и огромное количество транзисторов кэш-памяти, буферов команд, аппаратного предсказания ветвления и т.п. Графические процессоры используют транзисторы для массивов исполнительных блоков, блоков, управляющих потоками, разделяемой памяти небольшого объёма и контроллеров памяти на несколько каналов. Это не ускоряет выполнение отдельных потоков, оно позволяет графическому процессору обрабатывать несколько тысяч потоков, которые одновременно исполняются и требуют высокой пропускной способности памяти [21, 22].

Центральные процессоры используют кэш-память для увеличения производительности за счёт снижения задержек доступа к памяти (DRAM), а GPU используют кэш-память или общую память для увеличения полосы пропускания. CPU снижают задержки доступа к памяти при помощи кэш-памяти большого размера, а также предсказания ветвлений кода. Эти аппаратные части (Control) занимают большую часть площади кристалла микросхемы и потребляют много энергии.

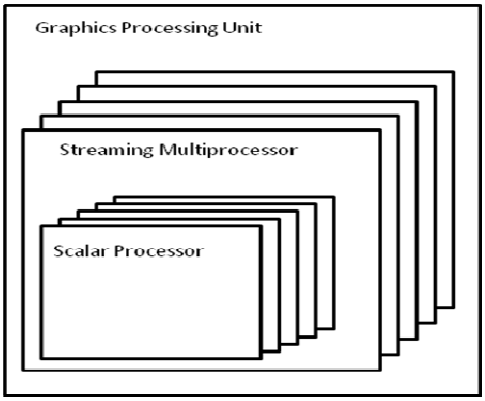

Рис. 6. Аппаратная организация графического процессора в программно-аппаратной модели CUDA

Современные подходы – увеличение количества вычислительных блоков в одном ядре и количества ядер в одном процессоре.

Маломощный процессор, исполнительный блок, либо ядро, имеют свою оперативную память и свою кэш-память [24]. Они объединяются в более крупные блоки. Так, потоковые процессоры объединяются в потоковые мультипроцессоры, которые также объединяют и другие исполнительные блоки (специализированные функциональные блоки и блоки вычислений с двойной точностью). Потоковые мультипроцессоры, в свою очередь, объединяются в кластеры текстурных процессоров, а они составляют уже целый графический процессор.

То есть, чем больше вычислительных блоков одновременно могут обрабатывать данные, тем выше степень распараллеливания и тем выше будет производительность (рис. 6).

5. Программная платформа для вычислений на GPU

Компания NVIDIA разработала программно-аппаратную платформу CUDA – технологию GPGPU с компилятором и библиотеками, позволяющую программистам реализовывать на языке программирования C алгоритмы, выполняемые на графических процессорах видеоадаптеров. Написание оптимального кода для видеоадаптеров задача нетривиальная и отличается от подходов к разработке программного обеспечения для CPU. CUDA раскрывает широкие возможности использования графического процессора и даёт программисту больший контроль над его аппаратными возможностями. Важно, что поддержка NVIDIA CUDA есть почти у всех современных видеоадаптеров компании [25]. Программно-аппаратная платформа CUDA доступна на 32-битных и 64-битных программных платформах Linux, Windows и MacOS X, т.е. охватывает практически весь спектр операционных систем.

Платформа CUDA является бесплатным комплектом средств разработки: (software development kit (SDK)) и доступна с сайта разработчика [26].

После создания и развития программно-аппаратной платформы CUDA разработчики получили возможность создавать приложения:

- моделирования сложных систем;
- обработки сигналов;
- вычислительной математики и геометрии;
- операций с базами данных;
- вычислительной биологии;
- вычислительной экономики;
- компьютерного зрения;
- системы обнаружения и предотвращения сетевых атак [27].

Каждый такт (clock cycle) GPU выполняет одни и те же инструкции над группой процессов, потоков (нитей). Различия между потоками GPU и CPU в том, что GPU-поток очень легковесен и не требует создания значительных накладных расходов. Для GPU необходимо создание нескольких тысяч потоков для достижения эффективности выполнения программы, а для CPU – только несколько потоков.

Поток в CUDA не имеет такой же смысл, как поток CPU. Поток GPU в данном случае является базовым набором данных, которые требуется обработать [28].

Warp в CUDA представляет собой группу из 32 потоков и является минимальным объёмом данных, обрабатываемых SIMD-способом в мультипроцессорах GPU.

В CUDA, вместо работы с warp напрямую, можно работать с блоками (Block), содержащими от 64 до 512 потоков, т.е. для облегчения работы с потоками в блок могут быть объединены максимум 16 warp. Эти блоки собираются вместе в сетки (Grid). Преимущество подобной группировки заключается в том, что число блоков, одновременно обрабатываемых GPU, тесно связано с аппаратными ресурсами. Если GPU имеет незначительное количество ресурсов, то он

будет выполнять блоки последовательно. Если число вычислительных процессоров велико, то блоки могут выполняться параллельно (рис. 7).

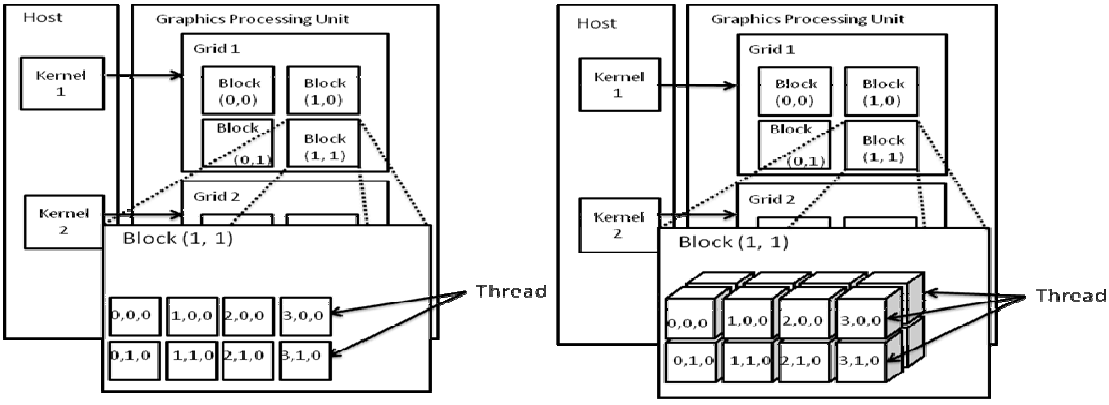

Рис. 1. Организация потоков в модели программирования программно-аппаратной платформы CUDA

Блоки потоков выполняются в виде warp. Количество потоков в одном warp – это размер warp, который может быть не больше 32. Графический процессор архитектуры G80 содержит 16 потоковых мультипроцессоров. Каждый потоковый мультипроцессор поддерживает максимум до 768 потоков. Каждый блок выполняется как warp с 32 потоками. В каждом потоковом мультипроцессоре исполняется до 8 блоков. Таким образом, в ситуации, когда активны только 256 потоков, может быть активно только 2 блока по 128 потоков либо 4 блока по 64 потока и т.д.

Для адресации каждого блока используется идентификатор этого блока, который может быть 2-мерным (например, (0,2)) либо одномерным, и идентификатор потока может быть как одномерным, 2- либо 3-мерным (например, (3,1,0)).

В терминологии CUDA термином хост (Host) называют CPU, устройство (Device) – обозначает GPU, программа будет – ядро (Kernel).

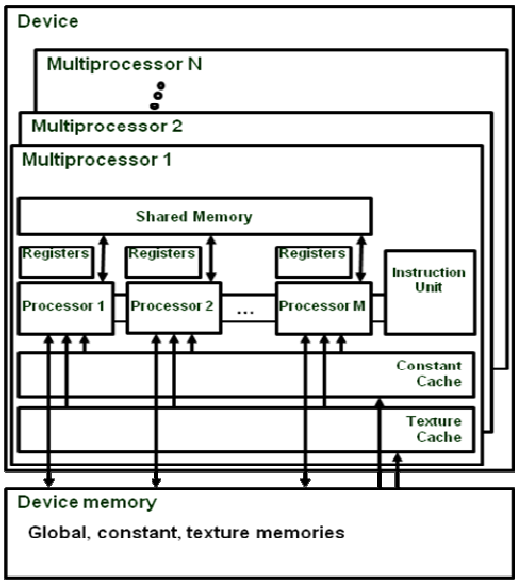

Рис. 8. Аппаратная модель памяти программно-аппаратной платформы CUDA

6. Организация памяти в программной платформе CUDA

Рассматривая потоки, необходимо отметить, какой вид доступа к памяти им предоставляется.

Регистровая память (register) является самой быстрой из всех видов. Все регистры GPU 32-разрядные. В CUDA нет явных способов использования регистровой памяти, всю работу по размещению данных в регистрах берет на себя компилятор (рис. 8).

Мультипроцессор использует 8192 (в архитектуре G80) либо 16384 (в архитектуре GT200) регистра, общих для всех потоков во всех блоках, выполняемых на нём. Максимальное число блоков на один мультипроцессор равно восьми, а число warp –

24 (768 потоков на один мультипроцессор). Всего видеоадаптеры архитектуры G80 могут обрабатывать до 12288 потоков одновременно, а видеоадаптеры на основе архитектуры GT200 – до 30720 потоков, т.к. на одном мультипроцессоре можно запустить до 1024 потоков (10 кластеров по три мультипроцессора, обрабатывающих 1024 потока). Знание этих ограничений позволяет оптимизировать алгоритмы под доступные ресурсы.

Локальная память (Local Memory) – это небольшой объём памяти, к которому имеет доступ только один потоковый процессор. Она относительно медленная, такая же, как и глобальная.

Разделяемая память(Shared Memory) – это 16-килобайтный блок памяти с общим доступом для всех потоковых процессоров в мультипроцессоре. Эта память весьма быстрая, такая же, как регистры. Она обеспечивает взаимодействие потоков и дает возможность обмена информацией между потоками в одном блоке, то есть все потоки в блоке выполняются одним мультипроцессором. Управляется разработчиком напрямую и имеет низкие задержки.

Преимущества разделяемой памяти:

• использование в виде управляемой программистом кэш-памяти первого уровня;

• снижение задержек при доступе потоковых процессоров к данным;

• сокращение количества обращений к глобальной памяти;

• отдельное обращение для каждой половины warp (half-warp). Как правило, требует явной синхронизации [29, 30].

Для снижения частоты обращения к этой памяти NVIDIA оснастила мультипроцессоры кэш-памятью (8 кбайт на мультипроцессор), хранящей константы и текстуры.

Память констант (Constant Memory) – область памяти объемом 64 Кбайт, доступная только для чтения всеми мультипроцессорами. Она кэшируется по 8 Кбайт на каждый мультипроцессор. Задержка этой памяти составляет несколько сот тактов при отсутствии нужных данных в кэш-памяти, поэтому считается медленной.

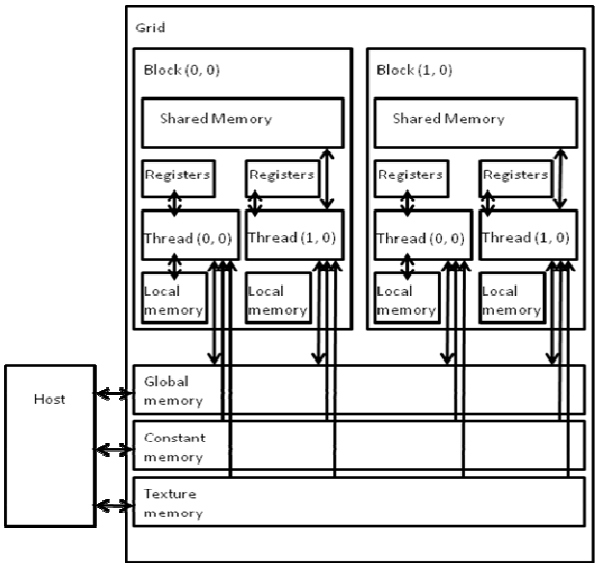

Рис. 9. Модель программирования: назначение различных типов памяти в одном мультипроцессоре с учетом взаимодействия с CPU

Текстурная память (Texture Memory) – блок памяти, доступный для чтения всеми мультипроцессорами. Выборка данных осуществляется при помощи текстурных блоков графического процессора, поэтому предоставляются возможности линейной интерполяции данных без дополнительных затрат. Кэшируется по 8 Кбайт на каждый мультипроцессор. Медленная, как глобальная, – сотни тактов задержки при отсутствии данных в кэш-памяти.

Глобальная память (Global Memory) – самый большой объём памяти, доступный для всех мультипроцессоров графического процессора, размер составляет от 256 Мбайт до 4 Гбайт на

видеоадаптере Tesla 1060 [31]. Значительно медленнее, чем разделяемая память, но доступна для чтения/записи сетке блоков.

Глобальная, локальная, текстурная и память констант – это физически одна и та же память, известная как локальная видеопамять видеоадаптера DRAM. Их отличия – в различных алгоритмах кэширования и моделях доступа. Центральный процессор может обновлять и запрашивать только внешнюю память: глобальную, константную и текстурную (рис. 9).

Общие характеристики различных типов памяти видеоадаптеров с поддержкой программно-аппаратной платформы CUDA можно увидеть в табл. 4.

Таблица 4. Характеристики типов памяти видеоадаптера

Память	Расположение	Доступ к памяти	Использование памяти	Задержки
Регистры	На кристалле GPU	Чтение/запись	Один поток	Один цикл
Локальная	На плате видеоадаптера	Чтение/запись	Один поток	Медленная
Разделяемая	На кристалле GPU	Чтение/запись	Все потоки в одном блоке	Один цикл
Глобальная	На плате видеоадаптера	Чтение/запись	Все потоки и CPU	Медленная
Память констант	На плате видеоадаптера	Чтение	Все потоки и CPU	До 100циклов
Текстурная	На плате видеоадаптера	Чтение	Все потоки и CPU	До 100циклов

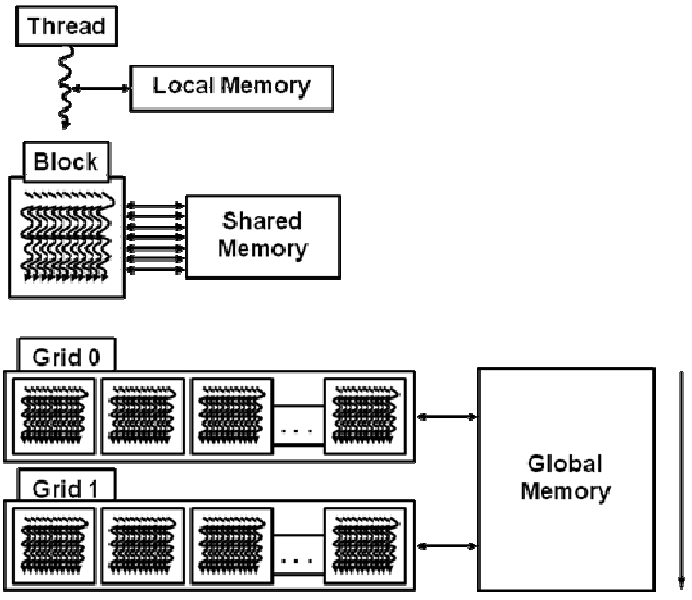

Рис. 10. Схематическое изображение связей с памятью сетки, блока и потока

Как уже упоминалось, мультипроцессор имеет 8192 регистра, общих для всех потоков, активных на мультипроцессоре.

Каждый поток может осуществлять:

• чтение/запись внутри регистров;

• чтение/запись внутри потоковой локальной памяти;

• чтение/запись внутри разделяемой памяти блока;

• чтение/запись внутри сетки в глобальной памяти;

• только чтение внутри сетки в памяти констант;

• только чтение внутри сетки в текстурной памяти.

CPU может обращаться к глобальной, текстурной и памяти констант (рис. 10) [32, 33].

Программист обязан тщательно рассчитывать размещение алгоритма в памяти – самые серьезные ограничения по знанию и пониманию программно-аппаратного комплекса CUDA для написания оптимального кода для GPU.

7. Интерфейс прикладного программирования платформы CUDA

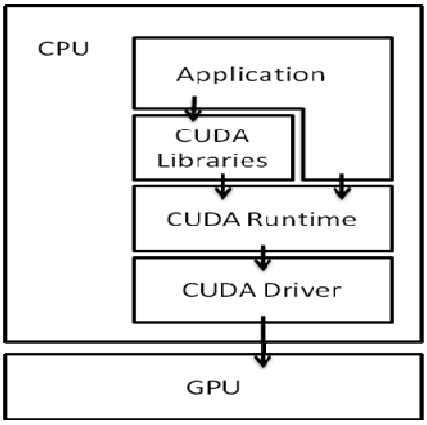

Рис. 11. Интерфейс прикладного программирования CUDA

CUDA включает два уровня API [30]:

• высокого (CUDA Runtime);

• низкого (драйвер, CUDA Driver).

Программист может использовать или один, или другой – в одной программе одновременное использование обоих невозможно.

Высокоуровневый API логически реализован над низкоуровневым, каждый вызов функции высокого уровня разделяется на более простые инструкции, которые обрабатывает драйвер. Для работы с API необходимы знания об устройстве и работе видеоадаптеров NVIDIA (рис. 11). Кроме того, возможно использование CUDA-библиотек:

• CUBLAS – CUDA – вариант, предназначенный для вычислений линейной алгебры (Basic Linear Algebra Subprograms(BLAS));

• CUFFT – CUDA – вариант библиотеки для расчёта быстрого преобразования Фурье (Fast Fourier Transform).

8. Выводы

Подходы к проблеме повышения производительности компьютеров в настоящее время меняются. Увеличение частот и количества транзисторов (экстенсивный путь) модифицирует в сторону увеличения количества ядер, что влечет за собой увеличение числа транзисторов и распараллеливание потоков вычислений.

В настоящее время сформировалась тенденция перехода к использованию GPU, который изначально был спроектирован для выполнения параллельных задач в режиме работы сопроцессора CPU. При этом возможен многократный прирост производительности компьютера с видеоадаптером.

Новая программно-аппаратная платформа CUDA предоставляет возможность использования значительных ресурсов видеоадаптера в целях, не связанных с обработкой изображения, например, для выполнения математических расчетов.

Для получения значительного увеличения быстродействия программы на GPU по сравнению с CPU программист должен досконально знать архитектуру графического процессора и правильно рассчитывать размещение алгоритма в памяти видеоадаптера. Это диктует специфика архитектуры графического процессора: неправильное использование памяти вызовет многочисленные циклы простоя GPU.

Производительность GPU значительно повышает производительность компьютера за счет большого числа потоковых процессоров, дающих возможность выполнять одновременно значительное количество потоков одной программы.

СПИСОК ЛИТЕРАТУРЫ

1. Moore G.E. Cramming more components onto integrated circuits / G.E. Moore // Electronics Magazine. – 1965. – N 8. – P. 114 – 117.
2. Multi-core – Wikipedia // Wikipedia – the free encyclopedia [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Multi-core_(computing).
3. Intel® Multi-Core Processors:Quick Reference Guide [Електронний ресурс] / T.W. Burger. – Режим доступу: http://www.intel.com/cd/ids/developer/emea/rus/261524.htm.
4. 60 years of the transistor: 1947 – 2007 [Електронний ресурс]. – Режим доступу: http://www.intel.com/technology/timeline.pdf.
5. GeForce GTX 295 Specifications [Електронний ресурс]. – Режим доступу: http://www.nvidia.com/object/product_geforce_gtx_295_us.html.
6. Intel® Atom™ Processor Microarchitecture [Електронний ресурс]. – Режим доступу: http://www.intel.com/technology/atom/microarchitecture.htm.
7. Shader – Wikipedia // Wikipedia – the free encyclopedia [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Shader.
8. Pixel Shaders [Електронний ресурс]. – Режим доступу: http://www.nvidia.com/object/feature_pixelshader.html.
9. CUDA Zone [Електронний ресурс]. – Режим доступу: http://www.nvidia.com/object/cuda_home.html.
10. ATI Stream Technology [Електронний ресурс]. – Режим доступу: http://www.amd.com/US/PRODUCTS/TECHNOLOGIES/STREAM-TECHNOLOGY/Pages/stream-technology.aspx.
11. Fatahalian K. A closer look at GPUs / K. Fatahalian, M. Houston // Communications of the ACM. – 2008. – N 10. – P. 50 – 57.
12. DirectX – Wikipedia // Wikipedia – the free encyclopedia [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/DirectX.
13. OpenGL – Wikipedia // Wikipedia – the free encyclopedia [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/OpenGL.
14. Hwu W.W. Programming Massively Parallel Processors. CUDA Threading-hardware [Електронний ресурс] / W.W. Hwu, D. Kirk // University of Illinois Urbana-Champaign. – 2009. – Режим доступу: http://courses.ece.illinois.edu/ece498/al/lectures/lecture8-threading-hardware-spring-2009.ppt.
15. OpenCL [Електронний ресурс]. – Режим доступу: http://www.khronos.org/opencl/.
16. Воеводин В. Графический вызов суперкомпьютерам / В. Воеводин, А. Адинец // Открытые системы. – 2008. – № 4. – С. 35 – 41.
17. Черняк Л. Многоядерные процессоры и грядущая параллельная революция / Л. Черняк // Открытые системы. – 2007. – № 4. – С. 34 – 42.
18. SIMD – Wikipedia // Wikipedia – the free encyclopedia [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/SIMD.
19. Мюллер С. Ремонт и модернизация ПК / С. Мюллер. – М., 2005. – 14-е изд. – С. 111 – 119.
20. Abi-Chahla F. Nvidia GeForce GTX 260/280 Review [Електронний ресурс] / F. Abi-Chahla, F. Charpentier. – Режим доступу: http://www.tomshardware.com/reviews/nvidia-gtx-280,1953.html.
21. Берилло А. NVIDIA CUDA – неграфические вычисления на графических процессорах [Электронный ресурс] / А. Берилло // Журнал IXBT. – 2003. – Режим доступа: http://www.ixbt.com/video3/cuda-1.shtml.
22. Tesla S1070 Specifications [Електронний ресурс]. – Режим доступу: http://www.nvidia.com/object/product_tesla_s1070_us.html.
23. FLOPS – Wikipedia // Wikipedia – the free encyclopedia [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/FLOPS.
24. Боресков А.В. Архитектура Tesla. Программно-аппаратный стек CUDA [Электронный ресурс] / А.В. Боресков, А.А. Харламов // Российское сообщество разработчиков CUDA. – 2009. – Режим доступа: http://cudacsmsusu.googlegroups.com/web/L2+03032009.zip?hl=ru&gda=tJnub0AAAACtimKUlGjG1mnXYmCE8iTpjTqzloEXGNYn22UUP5cnuoFr9eo7vgU__QTDrN9iHUidYk1usZk-V7uCv_MzusKC.
25. CUDA-Enadled Products [Електронний ресурс]. – Режим доступу: http://www.nvidia.com/object/cuda_learn_products.html.
26. Download CUDA [Електронний ресурс]. – Режим доступу: http://www.nvidia.com/object/cuda_get.html.
27. Gnort: High Performance Network Intrusion Detection Using Graphics Processors [Електронний ресурс] / G. Vasiliadis, S. Antonatos, M. Polychronakis et al. // Institute of Computer Science, Foundation for Research and Technology – Hellas. – 2008. – Режим доступу: http://www.ics.forth.gr/dcs/Activities/papers/gnort.raid08.pdf.
28. Hwu W.W. Programming Massively Parallel Processors. CUDA Memory fall [Електронний ресурс] / W.W. Hwu, D. Kirk // University of Illinois Urbana-Champaign. – 2009. – Режим доступу: http://courses.ece.illinois.edu/ece498/al/lectures/lecture5-cuda-memory-spring-2009.ppt.
29. Боресков А.В. Иерархия памяти CUDA [Электронный ресурс] / А.В. Боресков, А.А. Харламов // Российское сообщество разработчиков CUDA. – 2009. – Режим доступа:

http://cudacsmsusu.googlegroups.com/web/L4+17032009.zip?hl=ru&gda=6RTjrEAAAACtimKUlGjG1mnXYmCE8iTp52OoKPqrVGyghJAWbauLT0w5O1CUkDu5YWhIZ8OAPq2dYk1usZk-V7uCv_MzusKC.
30. Чеканов Д. NVIDIA CUDA: вычисления на видеокарте или CPU [Электронный ресурс] / Д. Чеканов. – Режим доступа: http://www.thg.ru/graphic/nvidia_cuda/onepage.html.
31. Tesla C1060 Specifications [Електронний ресурс]. – Режим доступу: http://www.nvidia.com/object/product_tesla_c1060_us.html
32. Hwu W.W. Programming Massively Parallel Processors: Memory Hardware in G80 [Електронний ресурс] / W. W. Hwu, D. Kirk // University of Illinois Urbana-Champaign – 2009. – Режим доступу: http://courses.ece.illinois.edu/ece498/al/lectures/lecture9-memory-hardware-spring-2009.ppt.
33. Scalable Parallel Programming with CUDA / J. Nickolls, I. Buck, M. Garland et al. // ACM Queue. – 2008. – Vol. 6, N 2. – С. 40 – 56.

Стаття надійшла до редакції 28.05.2009