электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.272

Перспективы использования графических процессоров для параллельной обработки данных

Р.А. Кладовщиков

Студент, МГТУ им. Н.Э. Баумана (Калужский филиал), г. Калуга, Россия

Научный руководитель: Ю.С. Белов,к.ф.-м.н,, доцент кафедры «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» МГТУ им. Н.Э. Баумана (Калужский филиал), г. Калуга, Россия

КФ МГТУ им. Н.Э. Баумана
kladroman@yandex.ru

В течение 30 лет одним из основных методов повышения производительности бытовых компьютеров было увеличение тактовой частоты процессора. Но, как видно из таблицы, динамика роста частот постепенно замедляется. Причина тому – ряд технологических ограничений: на потребляемую мощность и на тепловыделение, а также быстро приближающийся физический предел размера транзистора. Показательный случай неудачной гонки за большими частотами произошел в ноябре 2004 года, когда компания Intel была вынуждена отменить выпуск модели Pentium 4 с тактовой частотой 4ГГц из-за проблем с теплоотводом. Большие вентиляторы способны решить проблему, но они слишком шумные, что, естественно, не устраивает пользователя.

Год	Тактовая частота	Процессор
1978	4.77 MHz	Intel 8086
1985	33 MHz	Intel 80386
1993	60 MHz	Intel Pentium
2004	3.46 GHz	Intel Pentium 4
2006	2.333 GHZ	Intel Core Duo T2700
2007	3 GHz	Intel Core 2 Duo E6800
2008	3.33 GHz	Intel Core 2 Duo E8600

2009	3.06 GHz	Intel Core i7 950
2011	3.4 GHz	Intel Core i7 2600
2012	3.5 GHz	Intel Core i7 3770

Конечно, увеличение тактовых частот – не единственный способ повышения производительности. Введение специальных инструкций, как например MMX (MultiMedia eXtension) и SSE (Streaming SIMD Extension), позволило ускорить выполнение вычислительных операций в определенных классах задач.

Для увеличения производительности, необходимо выполнить как можно большее число инструкций параллельно. Для этого, начиная с процессоров Intel Pentium, появилось суперскалярное выполнение, обеспечивающее выполнение двух инструкций за такт, а Pentium Pro отличался внеочередным выполнением инструкций.

Увеличение количества ядер процессоров также оказывает влияние на производительность. В 2005 году, столкнувшись с ростом конкуренции на рынке и имея не так уж много вариантов выбора, ведущие производители CPU стали предлагать процессоры с двумя вычислительными ядрами вместо одного. В последующие годы эта тенденция продолжилась выпуском CPU с тремя, четырьмя, шестью и восьмью ядрами. Эта так называемая мультиядерная революция знаменовала колоссальный скачок в развитии рынка потребительских компьютеров.

Максимальное ускорение, которое можно получить от распараллеливания программы на N процессоров (ядер), дается законом Амдала (Amdahl Law):

$$S = \frac{1}{(1+P) + \frac{P}{N}}$$

В этой формуле P – это часть времени выполнения программы, которая может быть распараллелена на N процессоров. Как легко видно, при увеличении числа процессоров N максимальный выигрыш стремится к $\frac{1}{(1-P)}$. Таким образом, если вы можете распараллелить ¾ всей программы, то максимальный выигрыш составит 4 раза.

В современных ПК присутствуют представляющие из себя массивно-параллельные вычислительные устройства с очень высоким быстродействием и большим объемом собственной памяти (DRAM). Такие устройства называют графическими процессорами (GPU).

На рисунке 1 приведена схема, отражающая архитектурные отличия в организации центрального (CPU) и графического процессоров (GPU). Конечно, разница в архитектуре подразумевает и различия в принципах работы.

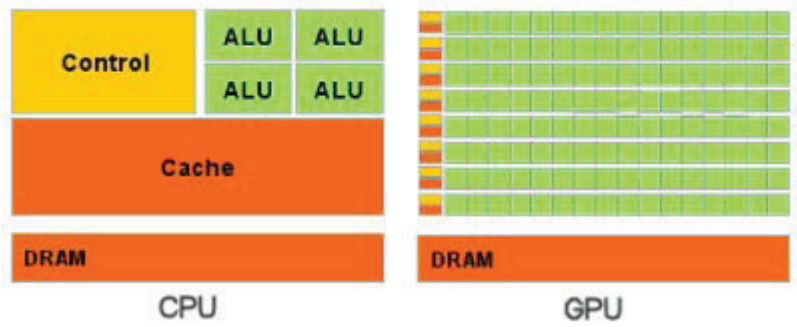

Рис. 1. Организация CPU и GPU

Например, отличается принцип доступа к памяти. В GPU он связанный и легко предсказуемый — если из памяти читается тексель текстуры, то через некоторое время придёт время и для соседних текселей. Да и при записи то же — пиксель записывается во фреймбуфер, и через несколько тактов будет записываться расположенный рядом с ним. Иными словами, это называется принцип локальности.

В универсальных процессорах большие количества транзисторов и площадь чипа идут на буферы команд, аппаратное предсказание ветвления и огромные объёмы начиповой кэш-памяти. Все эти аппаратные блоки нужны для ускорения исполнения немногочисленных потоков команд. Видеочипы тратят транзисторы на массивы исполнительных блоков, управляющие потоками блоки, разделяемую память небольшого объёма и контроллеры памяти на несколько каналов. Вышеперечисленное не ускоряет выполнение отдельных потоков, оно позволяет чипу обрабатывать нескольких тысяч потоков, одновременно исполняющихся чипом и требующих высокой пропускной способности памяти.

Видеочипы применяют SIMT (Single Instruction Multiple Thread, одна инструкция и несколько потоков) для скалярной обработки потоков. SIMT не требует, чтобы разработчик преобразовывал данные в векторы, и допускает произвольные ветвления в потоках.

На рисунке 2 представлены данные, отображающие превосходство графических устройств над центральными процессорами по количеству операций с плавающей запятой в секунду (floating-point operations per second, flops).

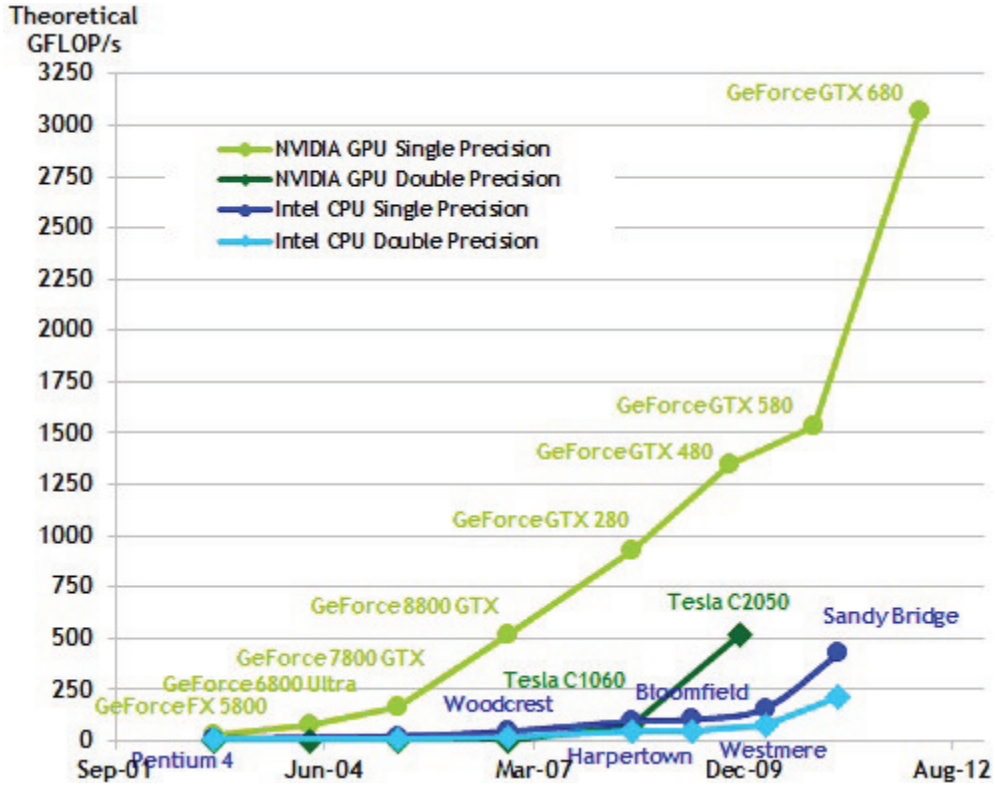

Рис. 2. График количества операций с плавающей запятой в секунду для устройств CPU и GPU

Такая производительность обеспечивается тем, что GPU состоит из нескольких потоковых мультипроцессоров, в который входят:

- Scalar Processors - скалярные процессоры
- Special Function Unit - блоки для выполнения специальных операций (к примеру, вычисления cos или sin)
- Instruction Fetch – модуль, отвечающий за инструкции (т.е. все скалярные процессоры выполняют одну команду)
- Register File – набор регистров
- Shared Memory – разделяемая память
- Double Precision – модуль для вычислений с двойной точностью.

На рисунке 3 приведена схема организации потокового мультипроцессора.

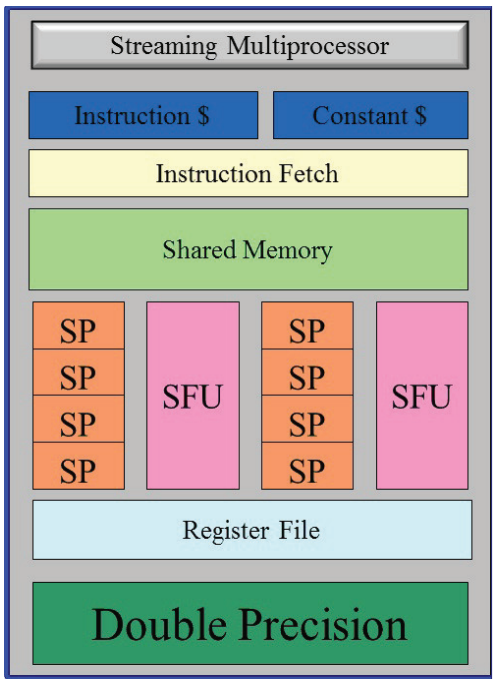

Рис. 3. Организация потокового мультипроцессора

Вполне очевидно, что данная схема отлично масштабируется. Увеличивая количество вычислительных блоков, можно повысить производительность. Например, GeForceGTX 680 содержит 1536 арифметико-логических устройств.

Конечно, высокая производительность GPU не могла не остаться без внимания. В результате возникло такое направление, как GPGPU (General-Purpose computing on Graphics Processing Units) – использование графических процессоров для решения неграфических задач. В качестве средств для GPGPU выступают CUDA, OpenCL и DX11 Compute Shaders.

В настоящее время ряд приложений, осуществляющих расчеты, уже обладают поддержкой вычислений на GPU. Например, в системе MATLAB реализовано более 200 функций, выполняемых на графическом процессоре. Ожидаемый прирост составляет 2-20 раз.

Вычисления на GPU предлагают беспрецедентную производительность приложений благодаря тому, что GPU обрабатывает части приложения, требующие большой вычислительной мощности, при этом остальная часть приложения выполняется на CPU. С точки зрения пользователя, приложение просто работает значительно быстрее.

CPU + GPU – это мощная комбинация, потому что CPU состоят из нескольких ядер, оптимизированных для последовательной обработки данных, в то время как GPU состоят из тысячи маленьких, более производительных ядер, созданных для параллельной

обработки данных. Последовательные части кода обрабатываются на CPU, а параллельные части - на GPU.

Список литературы

1. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. М.: ДМК Пресс, 2010. 232 с.
2. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах. М.: ДМК Пресс, 2011, 512 с.
3. Таненбаум Э. Архитектура компьютера. 5-е изд. СПБ.: Питер, 2007, 844 с.
4. Берилло А. NVIDIA CUDA – неграфические вычисления на графических процессорах // NVIDIA CUDA – неграфические вычисления на графических процессорах. URL.http://www.ixbt.com/video3/cuda-1.shtml (дата обращения: 06.03.2013).
5. Что такое вычисления на GPU? // Что такое вычисления на GPU? Высокопроизводительные вычислительные решения. NVIDIA URL.http://www.nvidia.ru/object/gpu-computing-ru.html (дата обращения: 06.03.2013).
6. CUDA C Programming Guide// CUDA C Programming Guide :: CUDA Toolkit Documentation URL.http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (дата обращения: 06.03.2013).