Д.А. Петрыкин (ЗАО «МЦСТ», МФТИ)

D. Petrykin

РАЗВИТИЕ ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ТИПА CPU-GPU

DEVELOPMENT OF CPU-GPU TYPE HETEROGENEOUS COMPUTING SYSTEMS

Рассмотрены архитектурные особенности современных гетерогенных вычислительных систем, включающих универсальные и графические процессоры.

The paper describes the architectural features of modern heterogeneous computing systems of the CPU-GPU type.

Ключевые слова: гетерогенные вычислительные системы, универсальные процессоры, CPU, графические процессоры, GPU.

Keywords: heterogeneous computing systems, central processing unit, CPU, graphics processing unit, GPU.

Введение

В гетерогенной, или гибридной вычислительной системе используются процессоры разных типов. Появление таких систем во второй половине XX века было связано преимущественно с успешной разработкой специализированных процессоров, ориентированных только на определенный класс задач, основным из которых стали графические приложения. Однако в наше время рост потребностей в более качественной обработке изображений и расширение технологических возможностей производства полупроводниковых кристаллов привели к тому, что графические процессоры стали быстро и основательно совершенствоваться, чем дали гетерогенным системам новое направление развития, состоящее в совместном использовании универсальных (CPU) и графических (GPU) процессоров в составе одной вычислительной системы при решении широкого класса задач. Принципиальная характеристика этой парадигмы приведена в разделе 1, а ее основные реализации – в разделе 2.

1. Универсальные вычисления с применением GPU

Изначально графические процессоры были разработаны исключительно для графических вычислений в режиме реального времени и имели довольно специфичную архитектуру, оптимизированную под конкретные алгоритмы обработки изображений. Однако по мере развития микропроцессорных технологий и появления потребностей в более сложной и качественной обработке визуальной информации в аппаратуре стали реализовывать все больше возможностей для программирования. В конечном счете, архитектура GPU обрела универсальность и стала поддерживать целый набор стандартных математических функций, что в некоторой степени приблизило модель программирования графики к модели программирования обычных пользовательских задач.

Современные GPU содержат сотни арифметических устройств и имеют высокую скорость доступа к внутренним и внешним модулям памяти, что позволяет им параллельно обрабатывать огромные массивы данных и достигать производительности в несколько TFlops. Все эти показатели, превосходящие почти на порядок аналогичные характеристики CPU, а также распространение параллельных вычислений и сложность масштабирования многоядерных и многопроцессорных систем заставили задуматься об использовании графического процессора для решения вычислительных задач, обладающих высоким уровнем параллелизма, и привели к возникновению нового направления вычислительной техники – General-Purpose computing on Graphic Processing Units (GPGPU) [1].

Эффективность GPGPU можно оценить с помощью гистограммы (рис. 1), опубликованной сотрудниками Калифорнийского университета по результатам исследования гетерогенной системы на задачах из пакетов Rodinia, Spec2006, Parsec и Spec2000 [2]. Столбцы гистограммы показывают процент общего программного кода, исполняемого на CPU. Как видно, за исключением некоторых задач, трудно совместимых с гетерогенной моделью, ее применение позволяет перенести на графический процессор исполнение от 50% до 99%

общего программного кода (на CPU остается в среднем по всем задачам 40,4%), а если учесть, что код исполняется на GPU на порядок быстрее, то время исполнения задачи этого набора на гетерогенной системе уменьшается минимум в два раза, а в самом оптимистичном случае – в десять раз.

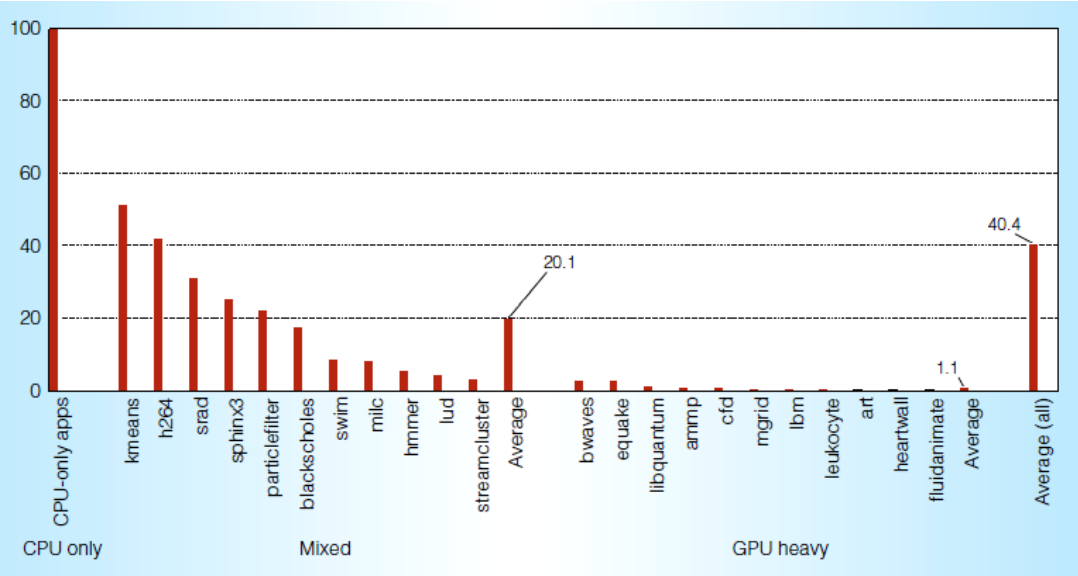

Рис. 1. Процент кода, исполняемый на CPU в составе гетерогенной системы

Несмотря на то что модель GPGPU показала существенное ускорение приложений, она имеет значительный недостаток. Дело в том, что GPU-программирование сильно отличается от CPU-программирования и имеет свою специфику, поэтому разработчик должен обладать глубокими знаниями графических API и архитектуры графического процессора, чтобы представить задачу в новой модели.

2. Основные реализации

NVIDIA G80 и CUDA

В 2007 году для решения проблем GPGPU-программирования корпорация NVIDIA представила новую архитектуру графических процессоров G80 и разработанную применительно к ней новую модель программирования CUDA (Compute Unified Device Architecture) [3].

Архитектура G80 принесла несколько ключевых нововведений, сильно упростивших

вычисления на GPU. Во-первых, отдельные вершинные и пиксельные конвейеры были заменены на единое и унифицированное арифметическое устройство, способное выполнять вершинные, геометрические, пиксельные и арифметические вычисления. Во-вторых, использование скалярных поточных графических ядер освободило программистов от необходимости ручного управления векторными регистрами. В-третьих, была реализована модель «single-instruction multiple-thread», в которой несколько независимых потоков выполняются параллельно с использованием одной команды. В-четвертых, для взаимодействия потоков стали использовать общую разделяемую память и барьерную синхронизацию.

В свою очередь, программная архитектура CUDA позволила разрабатывать программы для графических процессоров на C, C++, Fortran, OpenCL, DirectCompute и других языках высокого уровня. В этой модели параллельные участки программы исполняются на GPU как так называемые программные ядра (kernel). Каждое ядро состоит из большого набора параллельных потоков, которые должны быть очень «легкими», задействовать малое количество ресурсов и быстро переключаться. Взаимодействующие друг с другом потоки объединяются в блоки, в пределах которых они имеют общую разделяемую память и средства барьерной синхронизации (рис. 2). Массив таких блоков составляет сеть, которой выделяется участок глобальной памяти, используемый для получения исходных данных и выдачи результатов вычислений каждый раз после глобальной синхронизации всего ядра.

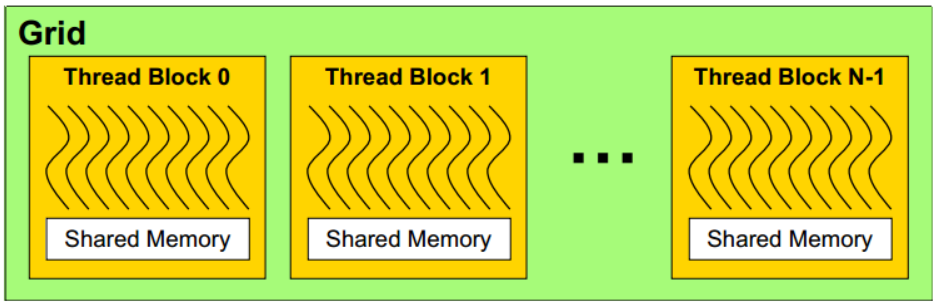

Рис. 2. Структура программного ядра в модели CUDA

Таким образом, корпорация NVIDIA, унифицировав и адаптировав под универсальные вычисления свои графические процессоры, а также предоставив для них более удобную модель программирования, позволила потребителям строить высокопроизводительные гетерогенные системы и эффективно использовать GPU для пользовательских задач с высоким уровнем параллелизма.

AMD Fusion

В AMD не стали придумывать собственные программные модели, а поддержали открытый фреймворк OpenCL, разработанный Khronos Group по спецификации Apple. Кроме этого, обладая своими графическими и универсальными процессорами, компания решила оптимизировать узкие места в аппаратуре для более эффективной работы в составе гетерогенной системы.

Вполне очевидно, что одним из узких мест гетерогенной системы является память. Даже, несмотря на десятикратное ускорений вычислений на GPU, все преимущества гетерогенной системы могут нивелироваться из-за медленной передачи исходных данных и результатов из CPU и обратно. Чтобы решить эту краеугольную проблему, AMD уже в первой декаде XXI века разработала технологию Fusion [4], позволяющую объединить центральный и графический процессоры внутри одного кристалла. Гетерогенные процессоры новой архитектуры получили название Accelerated Processing Unit (APU).

На рис. 3, где изображена структурная схема APU первого поколения под названием Llano, видно, что у графического ускорителя нет собственной физической памяти, а вместо нее используется участок оперативной памяти процессора. Передача графических данных в контроллер DDR3 и обратно происходит через специальный некогерентный интерфейс – шину Radeon Memory Bus (RMB), которая продублирована для каждого канала памяти и работает на тактовой частоте северного моста. Для доступа к разделяемым данным существует отдельный когерентный интерфейс – Fusion Control Link (FCL). На ри-

сунке перечисленные шины обозначены MEM PHY и IO PHYS.

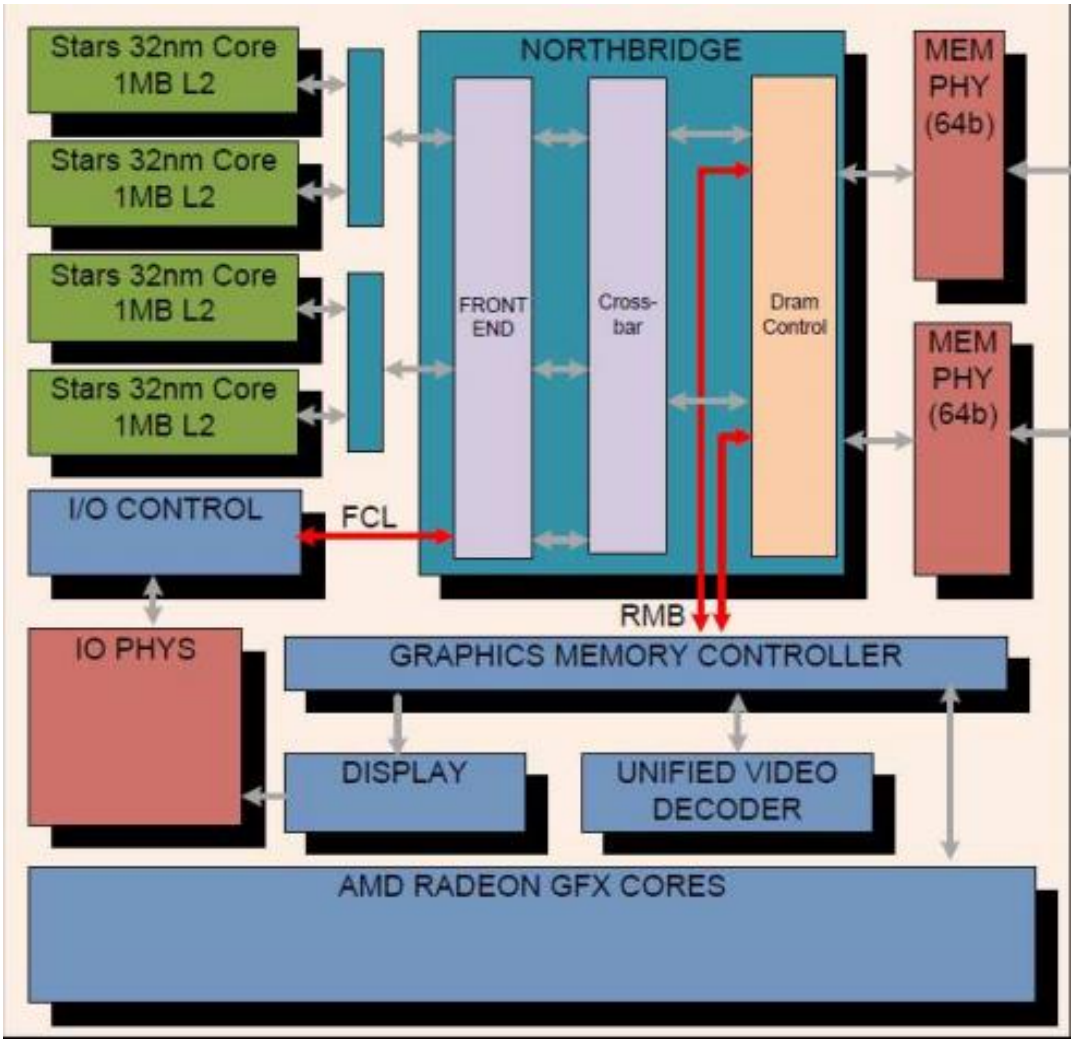

Рис. 3. Структура APU Llano

Стоит отметить, что архитектура APU подходит как для портативных вычислительных систем из-за пониженного энергопотребления и интеграции на одном кристалле, так и для небольших серверов за счет выигрыша GPGPU модели при использования общей физической памяти центральным и графическим процессорами.

Heterogeneous System Architecture (HSA)

Дальнейшее развитие APU получило в рамках концепции Heterogeneous System Architecture (HSA) [5], суть которой состоит в более тесной интеграции GPU и CPU. Как отмечалось ранее, для APU характерно, что графические и универсальные ядра обладают общей физической памятью. Однако их виртуальная память так и осталась раздельной,

поэтому программному обеспечению, управляющему гетерогенной системой, приходится по-прежнему копировать данные из одного фрагмента памяти в другой.

Концепция HSA устраняет этот недостаток, предлагая единое виртуальное пространство (рис. 4), позволяющее организовать обмен данными между GPU и CPU посредством передачи указателей на данные вместо того, чтобы их копировать.

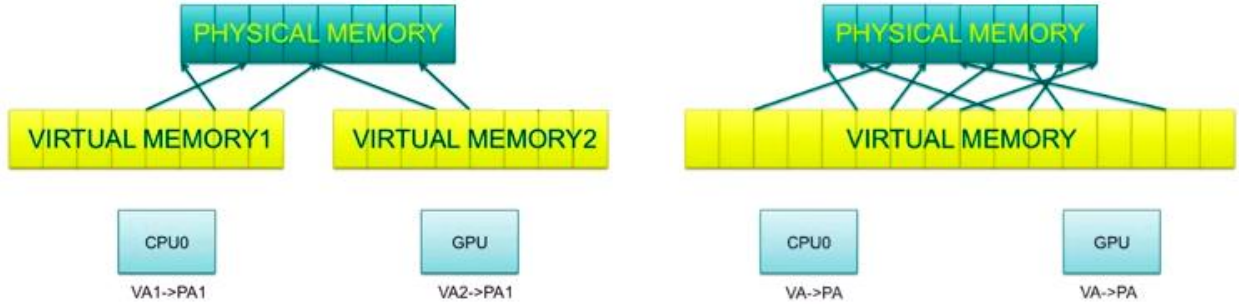

Рис. 4. Организация виртуальной памяти в APU первых поколений (слева) и в концепции HSA (справа)

Intel Sandy Bridge

Тот факт, что корпорация Intel не собирается отставать от конкурентов и тоже будет развивать гетерогенную систему, стал понятен уже с выпуском процессора Clarkdale с графическим ядром, реализованным на отдельном полупроводниковом кристалле.

Этот интеграционное решение было окончательно закреплено в следующей микроархитектуре – Sandy Bridge, где графическое ядро разместили в одном кристалле с центральным процессором (рис. 5). Несмотря на то что процессоры Sandy Bridge попадают под определение APU, введенное AMD, корпорация Intel по коммерческим причинам избегает этого термина и называет свое решение «центральным процессором с интегрированной графикой».

В отличие от AMD в Intel не стали вводить отдельные шины между графическим ядром и памятью, а вместо этого сделали дополнительный выход на кольцевую шину, которая уже применялась в восьмиядерных процессорах Nehalem-EX для связи универсальных ядер, кэша L3, контроллеров DDR3 и других системных блоков. Данное решение суще-

ственно увеличило скорость доступа графического ядра к разделяемым когерентным данным, но, в то же время, кольцо стало узким местом системы, т.к. активная работа с кэшем L3 будет мешать передаче больших объемов некогерентных данных и наоборот.

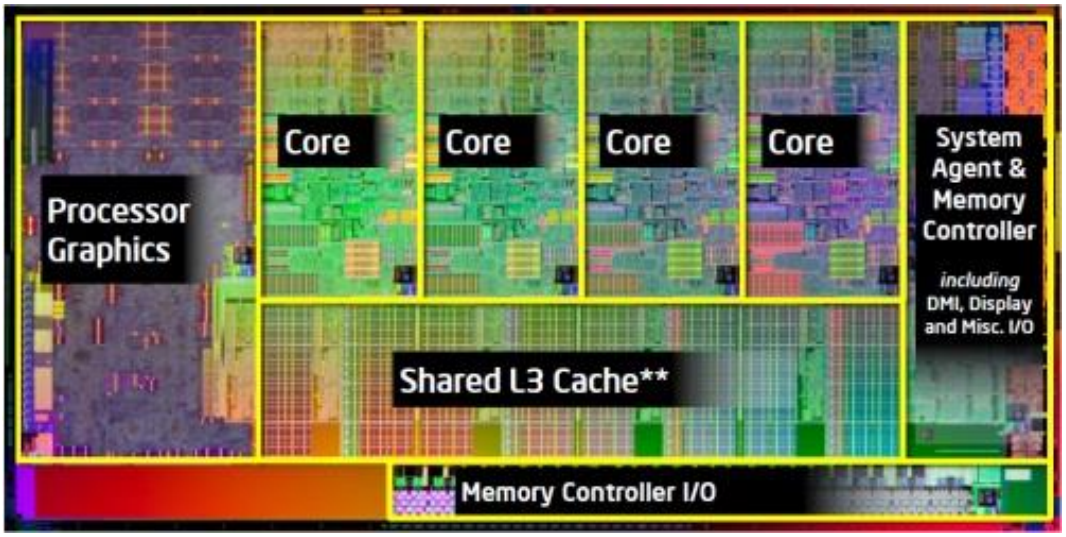

Рис. 5. Процессор микроархитектуры Sandy Bridge

Intel Xeon Phi

Отдельно от линейки процессоров с интегрированной графикой Intel разрабатывает сопроцессоры Xeon Phi с поддержкой модели GPGPU. Они предназначены преимущественно для многопоточных задач, обладая как векторным модулем, так и стандартной арифметикой x86 и x87. Можно сказать, что данный сопроцессор является чем-то средним между CPU и GPU, поэтому он будет проигрывать первому на непараллельных задачах, а второму – на векторных, но превзойдет обоих на «смешанных» приложениях, обладающих как параллельными, так и последовательными участками. Из графика на рис. 6 видно, что Xeon Phi становится эффективнее универсального процессора Xeon только для пятидесяти и более потоков [6].

Правильность данного решения была подтверждена тем фактом, что в июне 2013 года гетерогенный суперкомпьютер Tianhe-2, собранный из процессоров Xeon и сопроцессоров Xeon Phi, достиг производительности более 33 PetaFLOPS и возглавил рейтинг TOP500.

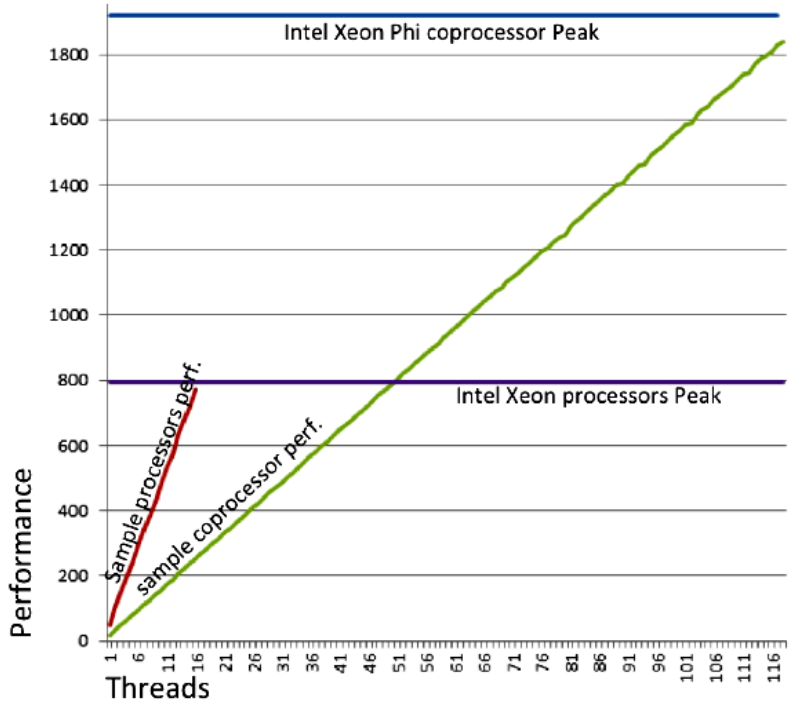

Рис. 6. Сравнение производительности Xeon и Xeon Phi

Заключение

Тенденция использования гетерогенных систем CPU-GPU типа постепенно набирает обороты и находит все большее распространение, несмотря на изначальное недовольство программистов. С одной стороны, постепенно совершенствуется программное обеспечение, появляются более удобные модели программирования и соответствующие высокоуровневые API. Но, с другой стороны, разработчики графических процессоров сделали большой шаг или даже разворот архитектуры в сторону универсальных вычислений.

Все эти факторы способствуют большему развитию гетерогенных систем, которые уже сейчас находят применение в мобильных, портативных, стационарных, серверных и даже суперкомпьютерных устройствах, превосходя аналоги по уровню параллелизма, энергопотреблению и даже производительности.

Литература

1. M. Rofouei, T. Stathopoulos, S. Ryffel, W. Kaiser, and M. Sarrafzadeh. Energy-Aware High Performance Computing with Graphic Processing Units. // In Proceedings of the 2008 conference on Power aware computing and systems, p. 11-11. USENIX Association, 2008.

2. M. Arora, S. Nath, S. Mazumdar, Scott B. Baden, Dean M. Tullsen. Redefining the role of the CPU in the era of CPU-GPU integration. – IEEE Micro, Nov.-Dec. 2012, Volume 32, Issue 6.

3. NVIDIA's Next Generation CUDA Compute Architecture: Fermi. – White Paper by NVIDIA Corporation, 2009.

4. D. Foley, P. Bansal, D. Cherepacha, R. Wasmuth, A. Gunasekar, S. Gutta, A. Naini. A Low-Power Integrated x86-64 and Graphics Processor for Mobile Computing Devices. – IEEE Journal Of Solid-State Circuits, Vol. 47, No. 1, January 2012.

5. G. Kyriazis. Heterogeneous System Architecture: A Technical Review. – AMD, 2012.

6. J. Reinders. An Overview of Programming for Intel Xeon processors and Intel Xeon Phi coprocessors. – Intel Corporation, 2012.