2. Классификация вычислительных систем: классификация Флинна. Архитектура многопроцессорных вычислительных систем: симметричная мультипроцессорность, одновременная многопотоковость, многоядерность, массивный параллелизм.

Среди всех рассматриваемых систем классификации ВС (вычислительных систем) наибольшее признание получила классификация, предложенная в 1966 году М. Флинном. В ее основу положено понятие потока, под которым понимается последовательность элементов, команд или данных, обрабатываемая процессором. В зависимости от количества потоков команд и потоков данных Флинн выделяет четыре класса ар­хитектур: SISD, MISD, SIMD, MIMD.

**SISD**

SISD (Single Instruction Stream/Single Data Stream) – одиночный поток команд и одиночный поток данных (рис. 2.1, а). Представителями этого класса являются, прежде всего, классические фон-неймановские ВМ, где имеется только один поток команд, команды обрабатываются последовательно, и каждая команда инициирует одну операцию с одним потоком данных. То, что для увеличения скорости обработки команд и скорости выполнения арифметических операций может применяться конвейерная обработка, не имеет значения, поэтому в класс SISD одновременно попадают как ВМ CDC 6600 со скалярными функциональными устройствами, так и CDC 7600 с конвейерными. Некоторые специалисты считают, что к SISD-системам можно причислить и векторно-конвейерные ВС, если рассматривать вектор как неделимый элемент данных для соответствующей команды.

**MISD**

MISD (Multiple Instruction Stream/Single Data Stream) – множественный поток команд и одиночный поток данных (рис. 2.1, б). Из определения следует, что в архитектуре ВС присутствует множество процессоров, обрабатывающих один и тот же поток данных. Примером могла бы служить ВС, на процессоры которой подается искаженный сигнал, а каждый из процессоров обрабатывает этот сигнал с помощью своего алгоритма фильтрации. Тем не менее ни Флинн, ни другие специалисты в области архитектуры компьютеров до сих пор не сумели представить убедительный пример реально существующей вычислительной системы, построенной на данном принципе. Ряд исследователей относят к данному классу конвейерные системы, однако это не нашло окончательного признания. Отсюда принято считать, что пока данный класс пуст. Наличие пустого класса не следует считать недостатком классификации Флинна. Такие классы, по мнению некоторых исследователей, могут стать чрезвычайно полезными для разработки принципиально новых концепций в теории и практике построения вычислительных систем.

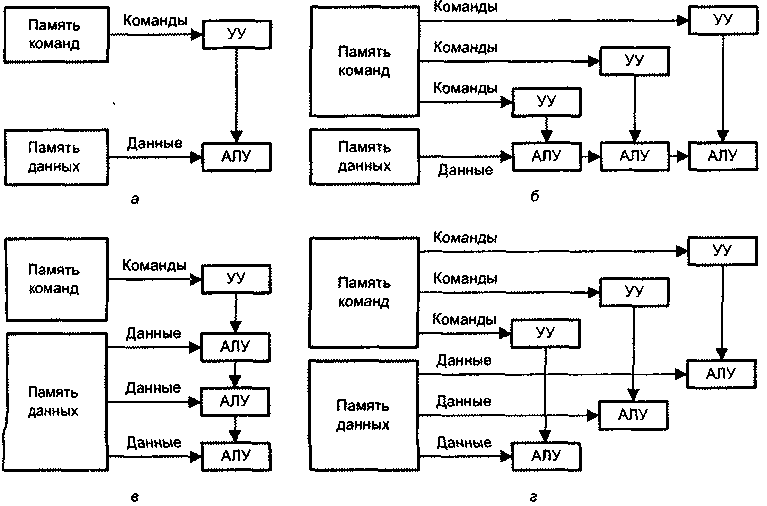


Рис. 2.1. Архитектура вычислительных систем по Флинну: а – SISD; б –MISD; в – SIMD; г – MIMD

**SIMD**

SIMD (Single Instruction Stream/Multiple Data Stream) – одиночный поток команд и множественный поток данных (рис. 2.1, в). ВМ данной архитектуры позволяют выполнять одну арифметическую операцию сразу над многими данными – элементами вектора. Бесспорными представителями класса SIMD считаются матрицы процессоров, где единое управляющее устройство контролирует множество процессорных элементов. Все процессорные элементы получают от устройства управления одинаковую команду и выполняют ее над своими локальными данными. В принципе в этот класс можно включить и векторно-конвейерные ВС, если каждый элемент вектора рассматривать как отдельный элемент потока данных.

**MIMD**

MIMD (Multiple Instruction Stream/Multiple Data Stream) – множественный поток команд и множественный поток данных (рис. 2.1, г). Класс предполагает наличие в вычислительной системе множества устройств обработки команд, объединенных в единый комплекс и работающих каждое со своим потоком команд и данных. Класс MIMD чрезвычайно широк, поскольку включает в себя всевозможные мультипроцессорные системы. Кроме того, приобщение к классу MIMD зависит от трактовки. Так, ранее упоминавшиеся векторно-конвейерные ВС можно вполне отнести и к классу MIMD, если конвейерную обработку рассматривать как выполнение множества команд (операций ступеней конвейера) над множественным скалярным потоком.

Схема классификации Флинна вплоть до настоящего времени является наиболее распространенной при первоначальной оценке той или иной ВС, поскольку позволяет сразу оценить базовый принцип работы системы, чего часто бывает достаточно. Однако у классификации Флинна имеются и очевидные недостатки, например неспособность однозначно отнести некоторые архитектуры к тому или иному классу. Другая слабость – это чрезмерная насыщенность класса MIMD. Все это породило множественные попытки либо модифицировать классификацию Флинна, либо предложить иную систему классификации.

19. Неграфические вычисления на графических процессорах: предпосылки, история возникновения, основные технологии. Архитектура графических процессоров. Архитектура NVIDIA CUDA. Иерархия памяти в CUDA.

Предпосылки

Центральные процессоры разрабатываются для получения максимальной производительности на потоке инструкций, которые обрабатывают разные данные (как целые числа, так и числа с плавающей запятой), производят случайный доступ к памяти и т. д. До сих пор разработчики пытаются обеспечить больший параллелизм инструкций - то есть выполнять как можно большее число инструкций параллельно.

Проблема заключается в том, что у параллельного выполнения последовательного потока инструкций есть очевидные ограничения, поэтому слепое повышение числа вычислительных блоков не даёт выигрыша, поскольку большую часть времени они всё равно будут простаивать.

История

Первые попытки использовать графические ускорители для нецелевых вычислений предпринимались еще с конца 90-х годов. Однако масштабная работа в этом направлении началась немного позже. В начале 2000-х годов два основных лидера в мире процессоров – компании AMD и Intel – решили побороться за производство самого мощного «мозга» компьютера. В период жесткой конкуренции производителям удалось значимо поднять планку обработки процессорами тактовых частот (например, в промежуток между 2001-м и 2003-м годом частоты подросли с 1,5 до 3 ГГц).

Вместе с этим разработчики из Стенфордского университета поставили перед собой конкретную цель – создать графический адаптер, который позволит проводить неграфические вычисления. Так появился Brook – специализированный язык программирования и реализации. Затем были запущены аналогичные проекты: библиотека Accelerator, Brahma, GPU++ и другие.

Немногим позже к исследовательской работе присоединился лидер индустрии NVIDIA. Компания решила разработать собственные программные платформы и видеокарты для неграфических вычислений. Так появилась CUDA.

Архитектура графических процессоров

Работа GPU относительно простая. Она заключается в принятии группы полигонов с одной стороны и генерации группы пикселей с другой. Полигоны и пиксели независимы друг от друга, поэтому их можно обрабатывать параллельно. Таким образом, в GPU можно выделить крупную часть кристалла на вычислительные блоки, которые, в отличие от CPU, будут реально использоваться.

GPU отличается от CPU не только этим. Доступ к памяти в GPU очень связанный – если считывается тексель, то через несколько тактов будет считываться соседний тексель; когда записывается пиксель, то через несколько тактов будет записываться соседний. Разумно организуя память, можно получить производительность, близкую к теоретической пропускной способности. Это означает, что GPU, в отличие от CPU, не требуется огромного кэша, поскольку его роль заключается в ускорении операций текстурирования. Всё, что нужно, это несколько килобайт, содержащих несколько текселей, используемых в билинейных и трилинейных фильтрах.

Архитектура NVIDIA CUDA

GPGPU (англ. General-purpose computing on graphics processing units) – это технология, которая позволяет использовать графический процессор GPU в операциях, которые обычно выполняет центральный процессор CPU. Например, в математических вычислениях. С помощью GPGPU можно использовать видеокарту для выполнения неграфических вычислений. При этом графический процессор будет работать не вместо центрального, а в качестве вычислительного блока.

CUDA является улучшенной вариацией GPGPU. Она позволяет работать на специальном диалекте – это значит, что программисты могут использовать алгоритмы, предназначенные для графических процессоров при обработке неспецифических задач. С помощью CUDA можно значимо ускорить процессы обработки. Например, можно сократить время ожидания конвертации видео до 20 раз, не задействуя основной процессор.

GPU – архитектура общего назначения для задач с параллелизмом данных:

* Скалярные RISC инструкции
* Все типы математических операций
* Глобальная память
* Разделяемая память

Архитектура CUDA базируется на взаимодействии между CPU и GPU процессорами. Это возможно благодаря работе шейдера – программы для исполнения видеокарты процессорами.

Взаимодействие происходит с помощью разноуровневых API. API на первом уровне – Runtime. Он позволяет сделать первичную разбивку задачи на потоки. Затем эти декомпозированные потоки переходят на второй уровень API – Driver. При этом работу API поддерживают библиотеки NVIDIA – Libraries CUBLAS (для алгебраических расчётов) и FFT (расчёт по алгоритму ускоренного вычисления Фурье).

Если говорить максимально упрощенно, то работа алгоритма CUDA выглядит следующим образом:

* Центральный процессор выделяет нужное количество памяти и отправляет её графическому процессору.
* Центральный процессор запускает ядро и также «делится» им с графическим.
* Графический процессор выполняет операции используя память и мощности ядра.
* После обработки данных центральный процессор принимает результаты.

Основные этапы CUDA-программы(обобщенно)

1. Хост выделяет нужное количество памяти на устройстве.
2. Хост копирует данные из своей памяти в память устройства.
3. Хост стартует выполнение определенных ядер на устройстве.
4. Устройство выполняет ядра.
5. Хост копирует результаты из памяти устройства в свою память.

Иерархия памяти в CUDA

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 19.1 – Типы памяти

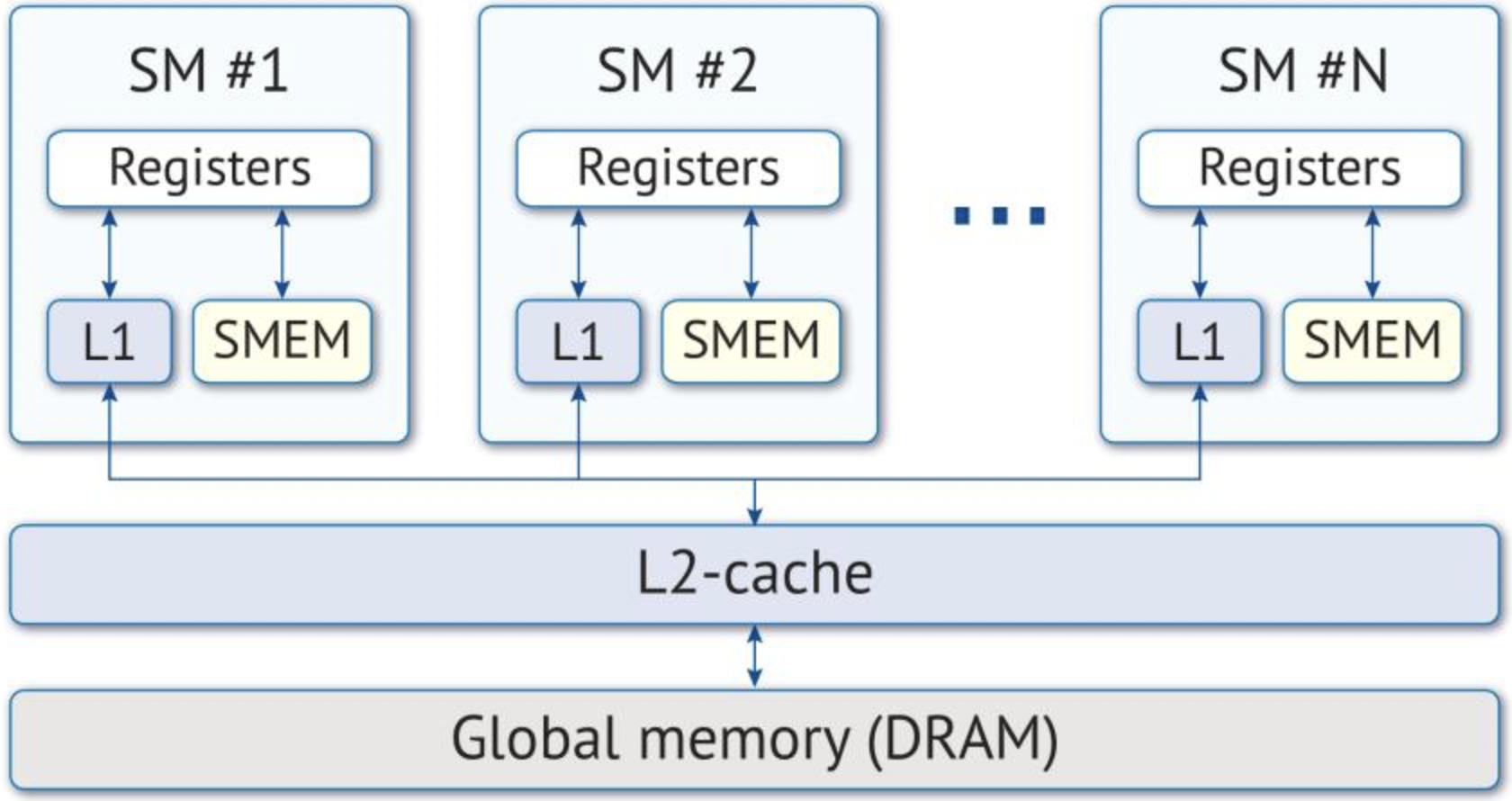


Рисунок 19.2 – Иерархия памяти

Иерархия памяти CUDA:

1. файл регистров
2. разделяемая память
3. кэш текстурный
4. кэш константный
5. кэш L1
6. кэш L2
7. глобальная память