Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

ВЫЧИСЛЕНИЯ НА ВИДЕОКАРТАХ

КУРСОВАЯ РАБОТА

студента 3 курса 351 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Григорьева Алексея Александровича

Научный руководитель
доцент _____ М. С. Семенов

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Будет добавлено позже. Немного об истории, сравнение характеристик процессора и видеокарты

1 Краткая теория

Составление эффективных алгоритмов вычисления на видеокарте в значительной степени отличается от привычных алгоритмов, исполняющихся на процессоре. При составлении программного кода необходимо учитывать как и общие особенности видеокарт, так и, возможно, характеристики конкретного устройства, для которого программируется алгоритм.

В данном разделе будет рассмотрена типовая модель видеокарты и основные понятия OpenCL, с которыми будем оперировать в данной работе.

1.1 Типовая модель видеокарты

Рассмотрим следующую архитектуру вычислительного устройства, используемого в видеокартах Nvidia 1.

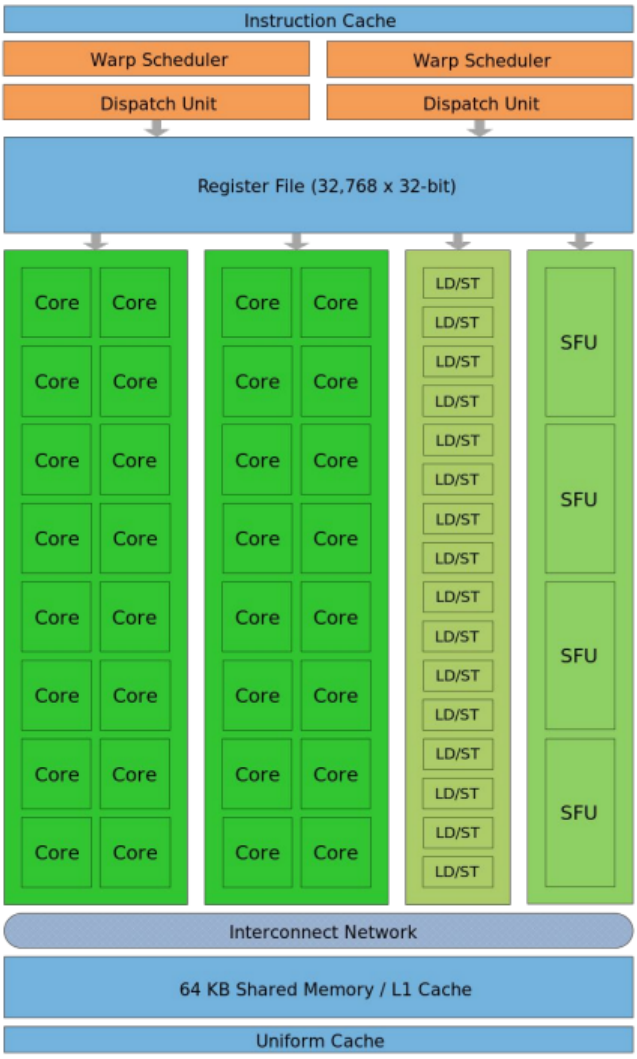

Рисунок 1 – Архитектура потокого мультипроцессора Fermi.

Вычислительное устройство в архитектуре Nvidia имеет 32 ядра (CUDA

cores), каждое из которых является потоком. В отличии от процессора, ядра выполняют более узкий набор задач, что позволяет с меньшими затратами увеличить их количество в устройстве. Для управления ими существует (warp scheduler), выполняющий роль указателя на инструкции соответствуя архитектуре SIMD. Данные для вычислений потоки берут из локальной памяти (shared memory), общей для всех ядер. Достигается это с использованием устройств загрузки и хранения (load-store units), соответственно способных загружать, а также сохранять данные в локальную память. Между всеми 32 ядрами динамически распределяются регистры, самая быстрая память, доступная им. У мультипроцессора в наличии намного больше регистров, чем могло быть нужно для выполнения программы. Это сделано для сокрытия времени на загрузку памяти и быстрого переключения контекста, подробнее - в разделе [???].

Количество таких устройств в видеокарте определяется следующим образом:

Количество ядер в видеокарте / 32, в случае Nvidia (warp)

Количество ядер в видеокарте / 64, в случае AMD (wavefront)

Например, видеокарта Nvidia Geforce GTX 1050 Ti имеет 768 ядер CUDA, и, соответственно, 24 warp.

1.2 Основные понятия OpenCL

OpenCL — открытый для свободного пользования программный интерфейс для создания параллельных приложений, использующих многоядерные структуры как и центрального процессора (CPU), так и графического (GPU). Использование API необходимо для обеспечения совместимости программы с различными устройствами.

При построении задач, определяется рабочее пространство (NDRange), представляющее все возможные в рамках задачи индексы потоков. Размер рабочего пространства определяется программистом на этапе инициализации OpenCL программы. Рабочее пространство может представлять:

- одномерный массив длиной N элементов;
- двумерную сетку размерности NxM;
- трехмерное пространство размерностью NxMxP.

Код, выполняющийся параллельно на ядрах процессора, называется kernel. Копия kernel выполняется для каждого индекса рабочего пространства и называется work-item с глобальным ID, соответствующим некоторому ID рабочего пространства. Kernel для всех work-item в рабочем пространстве имеют одинаковый код и входные параметры, но может иметь различный путь выполнения программы соответственно своему глобальному индексу - индекс в рабочем пространстве, полученному с использованием функции `get_global_id()`. Kernel в отличии от остальной программы полностью выполняется на видеокарте.

Группа work-item называется work-group, и за каждой группой закреплен собственный warp (см. предыдущий раздел), в рамках которого work-item могут синхронизироваться. Для каждой рабочей группы существует ее индекс в рабочем пространстве, и каждый work-item может узнать свой индекс внутри рабочей группы. Нетрудно заметить следующее соотношение:

$$\text{global ID} = \text{group ID} * \text{размер группы} + \text{local ID}$$

Размер рабочей группы аналогично рабочему пространству определяется программистом.

Каждое ядро, выполняя заданный kernel, является work-item в некоторой рабочей группой, на которые разделено рабочее пространство NDRange.

Рассмотрим на примере следующей схемы 2 другие виды сущностей, с которыми будет взаимодействие в дальнейшем.

- Платформа — драйвер, модель взаимодействия OpenCL и устройства. Распространены платформы от следующих производителей: Nvidia, Intel, AMD.
- Программа — хостовая часть, организующая подготовку к вычислениям и набор kernel-подпрограмм.
- Kernel — программа, исполняющаяся на видеокарте в каждом ядре.
- Контекст — окружение, в котором исполняется kernel.
- Объект памяти — создаваемый в контексте объект.
- Буфер — произвольный массив данных.

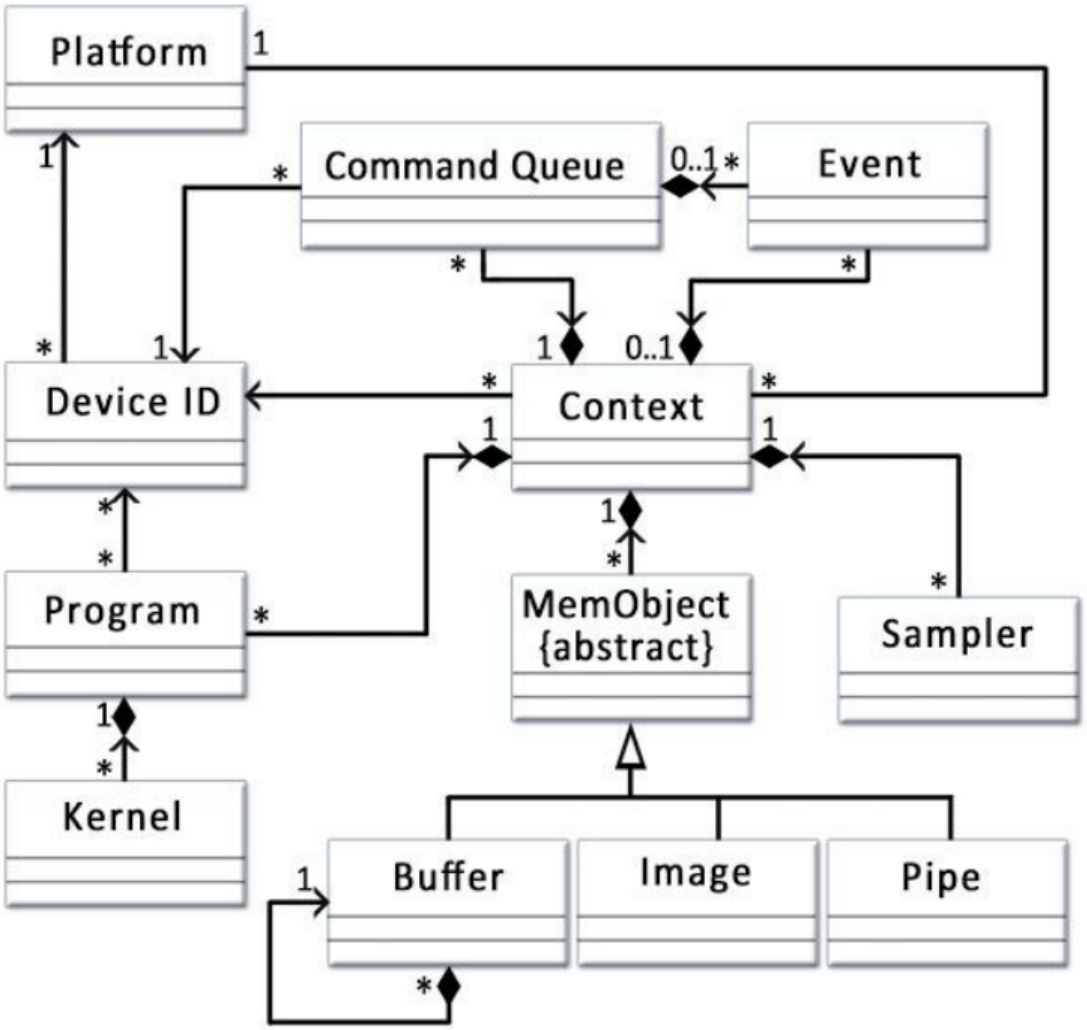

Рисунок 2 – Основные сущности в OpenCL.

2 Алгоритмы на видеокарте

В данном разделе будет рассмотрена анализ и практическая реализация алгоритмов на видеокарте, включая:

- описание общих требований к алгоритмам на основе доступа к памяти и параллельного исполнения;
- настройка среды разработки Visual Studio под выполнение параллельных программ с использованием OpenCL;
- написание программ для задач, использующих входные данные разных размерностей.

2.1 Требования к алгоритмам

Требования: наличие массового параллелизма; эффективный доступ к памяти; правильное ветвление и т.д 2 асимптотики, банк-конфликты, code divergence, register spilling, occupancy

2.2 Настройка среды разработки

CMake, VS, да и все

2.3 Инициализация OpenCL программы

Ну да, есть такое

2.4 Задачи на одномерных массивах

(сумма ряда, какая-нибудь префикс-функция)

2.5 Задачи на двумерных массивах

(транспонирование, переменожение матриц)

2.6 Задача на трехмерных массивах

(Collision detection)

ЗАКЛЮЧЕНИЕ

В настоящей работе

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А
Листинг программы

Код