МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Паралельнные алгоритмы»

Тема: Оптимизация доступа к памяти в модели OpenCL

Студент гр. 0304 _____ Максименко Е.М.

Преподаватель _____ Сергеева Е.И

Санкт-Петербург

2023

Цель работы.

Изучить способы оптимизации работы с памятью в OpenCL. С использованием OpenCL реализовать параллельное умножение матриц.

Задание.

Реализовать умножение матриц на OpenCL

В отчете: Произвести сравнение производительности с CPU реализацией из лаб. 4.

Выполнение работы.

1. Написание ядра (kernel).

Для вычисления произведения матриц используется OpenCL, поэтому необходимо написать kernel, производящий вычисления. Простой алгоритм умножения матриц требует многочисленного обращения к одним и тем же элементам матриц в процессе умножения. Так как обе матрицы хранятся в глобальной памяти, то обращение к элементам напрямую (доступ к глобальной памяти) — тяжелая операция. Для оптимизации доступа к памяти при умножении матриц обе матрицы разбиваются на квадратные ячейки (tiles). Ячейки для обеих матриц имеют одинаковые размеры. В локальной памяти создаются 2 буфера с размерами, равными размерам ячеек.

Каждая ворк-группа занимается умножением только своих ячеек. Идея умножения состоит в том, что для вычисления элементов результирующей матрицы необходимо умножать строки левой матрицы и столбцы правой. Каждая ворк-группа отвечает за одну строку ячеек левой матрицы и один столбец ячеек правой матрицы (см. рис. 1).

Произведение, как и в простом алгоритме, вычисляется путем перемножения строк левой матрицы и столбцов правой матрицы. Однако, так как ячейки левой и правой матриц имеют один размер, то для вычисления произведения можно посчитать сумму произведений строк ячеек левой

матрицы на столбцы ячеек правой матрицы, где индекс ячейки левой и правой матрицы совпадает (см. рис. 1). Таким образом, можно хранить в локальной памяти всего одну ячейку, а одна ворк-группа будет рассчитывать произведение для всех ячеек ей принадлежащих.

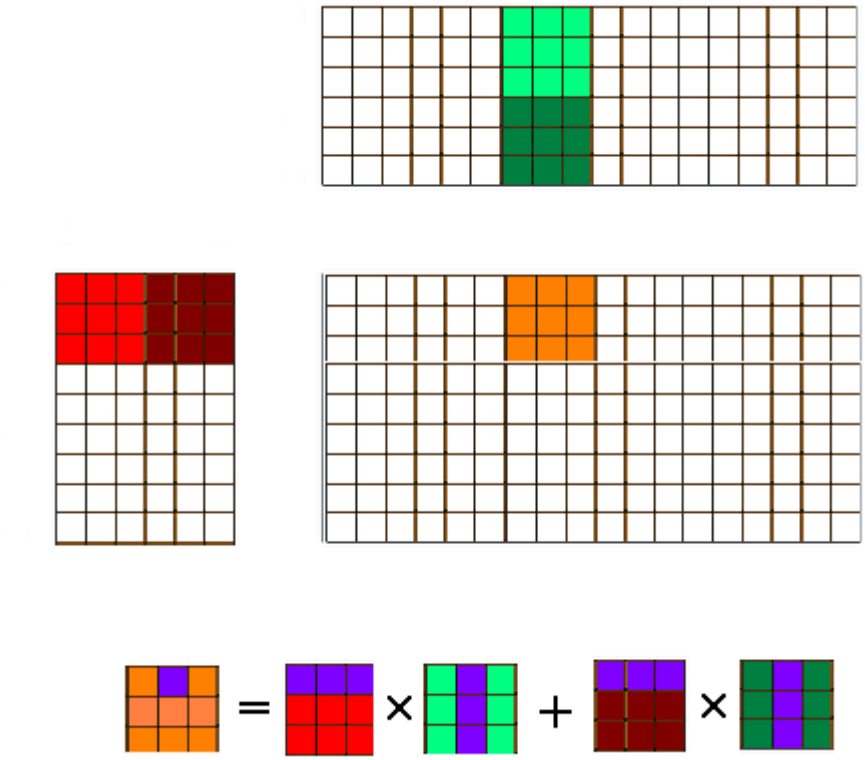

Рисунок 1. Разбиение на ячейки при умножении матриц

2. Написание программы хоста.

Для запуска расчетов с использованием OpenCL в программе хоста необходимо подготовить исполнение ядра. Для запуска ядра создается контекст OpenCL, происходит получение девайса, создание программы-ядра, создаются буферы, запускается ядро и считываются результаты (данные этапы описаны в прошлой работе более подробно).

Для разбиения матриц на ячейки необходимо, чтобы все размеры матриц делились нацело на размеры ячеек. В общем случае матрицы могут иметь любые размеры, поэтому для выполнения данного условия применяется выравнивание, добавленные для выравнивания элементы заполняются нулями. При таком выравнивании результаты умножения не

будут отличаться от результатов умножения без выравнивания (за исключением наличия нулевых строк/столбцов).

3. Замеры производительности.

Было проведено измерение времени выполнения данной реализации параллельного умножения и реализации алгоритма Штрассена. Результаты измерения времени выполнения программ см. в табл. 1.

Таблица 1. Время выполнения умножения матриц.

Размеры матриц	OpenCL GPU, мс	CPU алгоритм Штрассена, мс
128x128, 128x128	522	33
256x256, 256x256	475	121
512x512, 512x512	573	581
1024x1024, 1024x1024	626	3327
2048x2048, 2048x2048	948	21746

По табл. 1 видно, что для небольших матриц (до 512x512x512) время выполнения программы OpenCL сильно больше времени выполнения алгоритма Штрассена. Это связано с накладными расходами на подготовку программы OpenCL и обменом данными между ядром и устройством. Однако, при больших размерах матриц (более 512x512x512) время выполнения программы OpenCL становится сильно лучше времени выполнения алгоритма Штрассена. Также видно, что время выполнения программы OpenCL практически не зависит от размеров матриц (зависимость проявляется только при копировании данных между хостом и ядром).

Выводы.

В ходе работы были исследованы способы оптимизации доступа к памяти в модели OpenCL. Был реализован алгоритм умножения матриц с использованием OpenCL, в котором применена оптимизация с

использованием локальной памяти. Полученная реализация для больших размеров данных сильно превосходит алгоритм Штрассена.