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

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Параллельные алгоритмы»
Тема: Умножение матриц на GPU

Студентка гр. 1304 _____ Чернякова В.А.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2024

Цель работы.

Реализовать алгоритм умножения матриц с использованием Open CL.

Задание.

Реализовать блочное умножение матриц в стандарте OpenCL (или CUDA) с учётом оптимизаций доступа к локальной и глобальной памяти. Выполнить тестирование: сравнение результатов вычислений с полученными в работе 4. В отчете: произвести сравнение производительности с CPU реализациями сделанными в лаб. работе 4.

Выполнение работы.

Реализованная программа умножения матриц, состоит из 5 основных блоков:

1) Генерация входных матриц.

2) Выбор девайса.

3) Компиляция kernel-кода под выбранный девайс.

4) Запуск программы.

5) Сохранение результата в виде картинки.

Генерация входных матриц производится функцией generate_task, которая получает на вход размер матрицы и возвращает пару векторов целых чисел, заполненных случайными числами в пределах от -10 до 10.

Выбор девайса производится функцией create_device, которая посредством функций clGetPlatformIDs и clGetDeviceIDs выбирает нужный нам девайс и возвращает его id.

Компиляция kernel-кода производится функцией build_program, которая читает реализованный код из файла kernel.cl и посредством функций clCreateProgramWithSource и clBuildProgram собирает программу, готовую к исполнению, и возвращает ее.

Запуск программы осуществляется функцией invoke_kernel, которая перетает в собранный kernel-код аргументы с помощью функции clSetKernelArg,

запускает вычисления с помощью функции clEnqueueNDRangeKernel и по завершению вычислений переносит результаты из предоставленного буфера на host с помощью функции clEnqueueReadBuffer.

Сохранение результата производится функцией save_result, которая создает файл и записывает в него полученную матрицу, разделяя столбцы пробелами, а строки переводами строки.

Сравнение скорости умножения

Были проведены измерения времени от размеров вычисляемых матриц, результаты приведены в таблице. Количество потоков для простого и параллельного алгоритма 7.

Размер	**Простой алгоритм, мс**	**Параллельный алгоритм, мс**	**Алгоритм Штрассена, мс**	**OpenCL, мс**
16 × 16	1.51015	0.55202	0.116481	2.2569
32 × 32	0.655445	0.579161	0.685582	2.5383
64 × 64	2.56756	1.1307	1.46393	2.4747
128 × 128	9.04588	4.31222	11.4782	3.0438
256 × 256	65.2177	35.6696	31.7894	3.4578
512 × 512	615.772	334.645	237.683	7.1024
1024 × 1024	4918.72	2668.66	1796.51	45.1313
2048 × 2048	43059.5	22126.1	12056.3	346.613
4096 × 4096	332325	165552	81226	2873.48

По таблице видно, что для небольших матриц время вычисления на OpenCL больше. Это связано с загрузкой данных и созданием контекста OpenCL. По сравнению с этим временем, время самого вычисления пренебрежимо мало.

Для больших матриц время вычисления возрастает, но остаётся намного меньшим, чем время вычисления с помощью алгоритмов, использованных в 4 лабораторной работе.

Матричные вычисления с помощью OpenCL на GPU быстрее, потому что GPU обеспечивает параллельное выполнение на тысячи потоков, имеет быструю память и высокую пропускную способность, что делает его гораздо более эффективным для выполнения таких вычислений по сравнению с традиционными процессорами.

Выводы.

В ходе работы был реализован алгоритм умножения матриц с использованием OpenCL. При реализации были использованы оптимизации доступа к памяти.