**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**

| Студент гр. 1303 |  | Чубан Д.В. |
| --- | --- | --- |
| Преподаватель |  | Сергеева Е.И. |

Санкт-Петербург

2024

**Цель работы.**

Изучить механизмы доступа к памяти в модели CUDA.

**Задание.**

* Реализовать умножение матриц на CUDA
* Произвести сравнение производительности с CPU реализацией из лаб. 4.

**Выполнение работы.**

Для умножения матриц при помощи CUDA был написан kernel-код. Т.к. при классическом умножении матриц требуется многократный доступ к одним и тем же элементам, то в kernel-коде необходимые ячейки матрицы были сохранены в общей памяти, т. к. работа с глобальной памятью требует больших затрат. Для оптимизации умножения матрицы разбиваются на подматрицы одинакового размера.

Каждый поток (thread) внутри блока работает с одной элементарной

ячейкой результата умножения. Для этого в shared memory копируются

подматрицы из исходных матриц A и B. Эти подматрицы затем используются для вычисления произведения строк и столбцов с помощью итераций по всем блокам данных.

При умножении матриц необходимо, чтобы их размеры нацело делились на размеры подматриц, поэтому в случае, если это условие не выполняется, матрицы дополняются нулями, что не влияет на итоговый результат.

Были произведены замеры времени, необходимого для умножения матриц различного размера (размеры левой и правой матриц одинаковы) на CPU и на GPU.

| Размерность матрицы | Умножение на GPU | Масштабируемое параллельное умножение | Алгоритм Штрассена |
| --- | --- | --- | --- |
| 16x16 | 0.806463 | 0.000993409 | 0.000160345 |
| 64x64 | 0.386096 | 0.00184807 | 0.00326734 |
| 256x256 | 0.380596 | 0.0411845 | 0.0754281 |
| 1024x1024 | 0.425407 | 2.01045 | 3.00516 |
| 2048x2048 | 0.520602 | 15.4246 | 21.0118 |

По таблице видно, что при небольших размерах умножаемых матриц

алгоритмы на CPU оказываются быстрее, что обусловлено временными

затратами на подготовку к обработке кода и передаче данных в GRAM, но при росте размеров эти алгоритмы оказываются медленнее алгоритма на CUDA, при этом использование параллельных вычислений на GPU позволило ускорить вычисление произведения больших матриц в 30 раз.

**Вывод.**

Была изучена работа с глобальной и локальной памятями в CUDA.

Была реализована программа, умножающая матрицы параллельно при помощи CUDA.

Было произведено сравнение временных затрат на умножение при различных размерах матриц при помощи алгоритмов Штрассена и параллельного умножения матриц на CPU и при помощи CUDA. Было установлено, что при небольших размерах, алгоритмы на CPU работает быстрее, т. к. не требует передачи данных в GRAM, но при увеличении размеров алгоритм на CUDA оказывается быстрее.