МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №4**

**по курсу «Программирование графических процессоров»**

***Работа с матрицами. Метод Гаусса.***

Выполнил: А.Ю. Гришин

Группа: 8О-408Б

Преподаватель: А.Ю. Морозов

Москва, 2024

**Условие**

1. **Цель работы:** использование объединения запросов к глобальнйо памяти. Реализация метода Гаусса с выбором главного элемента по столбцу. Ознакомление с библиотекой алгоритмов для параллельных расчетов Thrust. Использование двухмерной сетки потоков. Исследование производительности программы с помощью утилиты nvprof.
2. **Вариант 2:** вычисление обратной матрицы.

**Программное и аппаратное обеспечение**

* Графический процессор
  + Compute capability: 7.5
  + Объем графической памяти: 15,83 ГБ
  + Объем постоянной памяти: 65536 байт
  + Разделяемая память на блок: 49152 байт
  + Количество регистров на блок: 65536
  + Максимальное количество потоков на блок: 1024
  + Количество мультипроцессоров: 40
* Процессор
  + Количество физических ядер: 2
  + Количество логических ядер: 4
  + Частота: 2000 МГц
* Оперативная память
  + Тип оперативной памяти: DDR4
  + Объем: 32 ГБ
* Жесткий диск
  + Объем: 1 ТБ
* Программное обеспечение
  + ОС: Ubuntu 22.04.4 LTS
  + IDE: Visual Studio Code
  + Компилятор: nvcc

**Метод решения**

Решение данной задачи я начал с определения алгоритма вычисления обратной матрицы с использованием метода Гаусса. Пусть A – исходная матрица. Перед использованием метода Гаусса необходимо построить блочную матрицу .

На следующем этапе, который называется прямым ходом, согласно методу Гаусса, используя элементарные преобразования, необходимо привести левую часть блочной матрицы к верхнетреугольной матрице.

После данного этапа происходит обратный ход. На этом этапе необходимо повторно выполнить метод Гаусса, однако, в этот раз в обратном направлении. В результате получится блочная матрица, левый компонент котороый представляет собой диагональную матрицу.

Заключительным этапом является нормализация значений. Для этого необходимо выполнить элементарное преобразование для каждой строки, разделив элементы правой компоненты на соответствующие значения диагональных элементов.

Метод Гаусса состоит из двух основных этапов на каждом шаге:

1. Вычисление коэффициентов;
2. Обновление значений элементов ниже текущей строки.

**Вычисление коэффициентов**

На произвольном шаге метода, когда текущая строка имеет номер , для всех строк с номерами вычисляются коэффициенты по следующей формуле:

Эти коэффициенты используются для последующего обновления значений элементов в строках ниже текущей. Поскольку вычисление каждого коэффициента зависит только от значений элементов текущей строки и не зависит от других коэффициентов, данный этап можно распараллелить.

Для этого этапа достаточно использовать одномерную сетку потоков, так как вычисление каждого коэффициента требует работы с данными, относящимися к одной строке, и не требует доступа к другим строкам.

**Обновление значений элементов**

После вычисления коэффициентов происходит обновление значений элементов строк ниже текущей по формуле:

Каждое новое значение элемента зависит только от вычисленных ранее коэффициентов, что также позволяет распараллелить вычисления. Для этого этапа целесообразно использовать двумерную сетку потоков, так как каждому потоку необходимо работать с элементами конкретной строки и столбца, то есть с элементами матрицы в двумерной форме.

**Оптимизация точности вычислений**

При вычислении коэффициентов методом Гаусса может возникнуть ошибка, связанная с делением на малые значения знаменателя, что приводит к значительной потере точности. Чтобы уменьшить эту погрешность, можно использовать оптимизацию путем выбора строки с наибольшим по модулю элементом в качестве ведущей строки на каждом шаге.

Таким образом, на каждом шаге метода выбирается строка с максимальным элементом по модулю в текущем столбце, что позволяет обеспечить максимально возможное значение знаменателя и, следовательно, повысить точность вычислений.

**Описание программы**

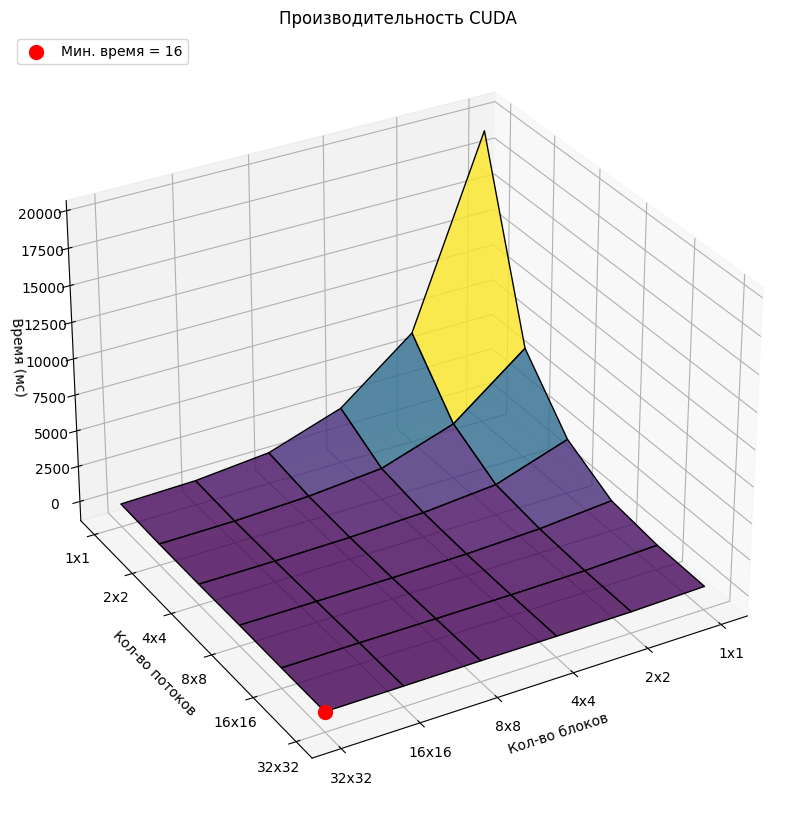
Программа состоит из одного файла, содержащего код, написанный на языке CUDA. Код программы состоит из:

1. Макроса EXIT\_WITH\_ERROR, который отвечает за вывод сообщения об ошибке в stderr и экстренный выход из программы с кодом возврата 0;
2. Макроса SAVE\_CUDA, который проверяет, что CUDA функция сработала успешно. В противном случае происходит экстренный выход через EXIT\_WITH\_ERROR;
3. Функции-ядра swapRows, которая отвечает за смену двух строк матрицы местами;
4. Функции-ядра initCoefs, которая отвечает за вычисление коэффициентов на каждом шаге прямого хода метода Гаусса;
5. Функции-ядра updateRows, которая отвечает за вычисление новых значений элементов на каждом шаге прямого хода метода Гаусса;
6. Функции-ядра initCoefsReversed, которая отвечает за вычисление коэффициентов на каждом шаге обратного хода метода Гаусса;
7. Функции-ядра updateRowsReversed, которая отвечает за вычисление новых значений элементов на каждом шаге прямого хода метода Гаусса;
8. Функции-ядра mult, которая выполняет умножение элементов строки матрицы на переданный коэффициент;
9. Функции-ядра initEyeMatrix для инициализации единичной матрицы размера ;
10. Типа Comparator, содержащего логику по сравнению элементов, что необходимо при поиске максимального по модулю элемента в матрице.

**Результаты**

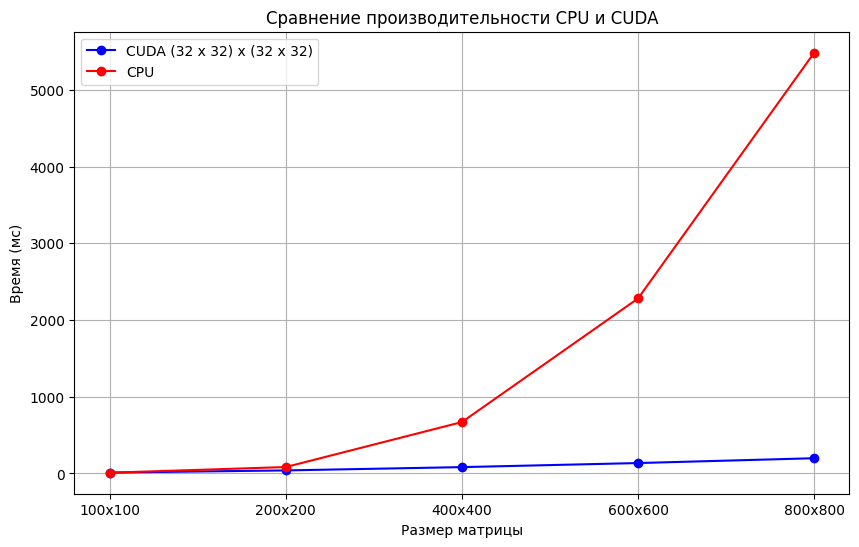
* + - 1. Зависимость времени выполнения программы от количества используемых потоков. В качестве единиц измерения времени были выбраны миллисекунды (мс). Тестирование производилось на матрице с размерами .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Количество блоков / потоков |  |  |  |  |  |  |
|  | 19255 | 6483 | 2536 | 830 | 353 | 232 |
|  | 6433 | 2501 | 817 | 368 | 164 | 119 |
|  | 2492 | 783 | 342 | 161 | 89 | 79 |
|  | 771 | 346 | 157 | 87 | 71 | 71 |
|  | 347 | 156 | 87 | 72 | 72 | 34 |
|  | 244 | 113 | 80 | 72 | 35 | 16 |

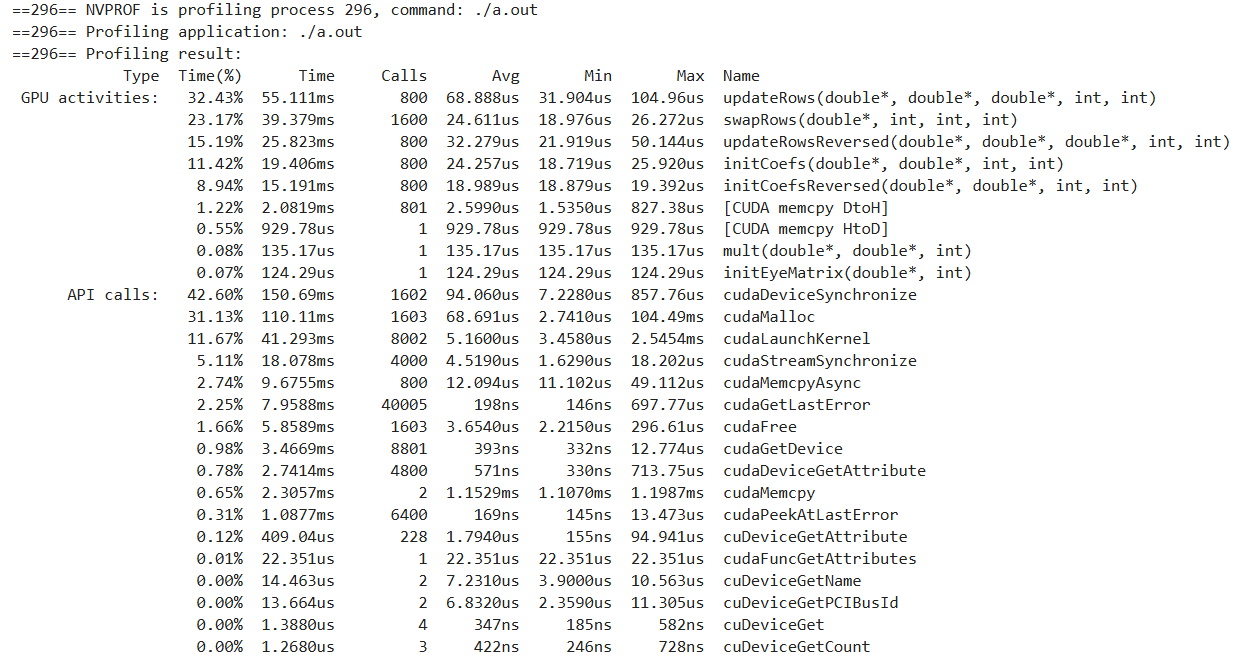


* + - 1. Сравнение программы на CUDA с блоками и потоками и программы на CPU с одним потоком. В качестве единиц измерения времени были выбраны миллисекунды (мс).

|  |  |  |
| --- | --- | --- |
| Размер матрицы | Время на CUDA (мс) | Время на CPU (мс) |
|  | 12 | 10 |
|  | 40 | 84 |
|  | 83 | 670 |
|  | 136 | 2281 |
|  | 199 | 5484 |



1. Результат исследования производительности с помощью nvprof (матрица , сетка потоков )



**Выводы**

В данной лабораторной работе был реализован алгоритм нахождения обратной матрицы методом Гаусса с выбором главного элемента по столбцу на языке CUDA.

В реализации алгоритма были применены оптимизации для эффективного использования глобальной памяти GPU, такие как объединение запросов к глобальной памяти. Данный подход позволил сократить общее количество обращений к памяти и повысить итоговую производительность программы. Основные вычислительные процессы распределялись с использованием двухмерной сетки потоков, что позволило более удобно отобразить вычисляемые элемены на потоки.

Для поиска максимального элемента по модулю в текущем столбце применялась библиотека Thrust, что упростило реализацию этой операции и обеспечило эффективный параллельный поиск.

Алгоритм был структурирован с помощью нескольких функций, каждая из которых выполняет отдельный этап, включая выбор главного элемента, вычисление коэффициентов для преобразования строк и нормализации элементов матрицы. Такая декомпозиция позволила выполнять некоторые действия алгоритма параллельно, при этом сохраняя общее последовательное выполнение.

Для оценки производительности программы была использована утилита nvprof. С ее помощью было измерено время выполнения отдельных ядер и анализированы узкие места, связанные с доступом к памяти.

Таким образом, в ходе лабораторной работы были изучены возможности эффективного использования памяти, а также были использовани инструменты для анализа производительности программы, таких как nvprof, для достижения более высокой эффективности вычислений на графических процессорах.