Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

ОТЧЁТ О ПРАКТИЧЕСКОЙ РАБОТЕ №4

по дисциплине
«Гибридные вычислительные системы»

Преподаватель _____ С. А. Тарасов
подпись, дата инициалы, фамилия

Студент КИ22-07Б, 032212677 _____ Л. А. Глушков
номер группы, зачетной книжкой подпись, дата инициалы, фамилия

Студент КИ22-07Б, 032215583 _____ А. М. Коробков
номер группы, зачетной книжкой подпись, дата инициалы, фамилия

Студент КИ22-07Б, 032214653 _____ И. О. Бердин
номер группы, зачетной книжкой подпись, дата инициалы, фамилия

Красноярск 2025

СОДЕРЖАНИЕ

Задание

1. Разработать кёрнел kernel_matmul_wmma, который принимает объекты MatrixView по значению и вычисляет произведение матриц, используя CUDA WMMA (см. рис. 1).

2. Перегрузить оператор operator* для класса Matrix, используя указанный кёрнел.

3. Используя фреймворк Google Test, разработать модульные тесты для operator* со следующими размерами матриц: A (m×k) и B (k ×n), где m,n,k ∈ {16,32,64,128,256,512}. В качестве эталона для сравнения исполь зовать результат аналогичной операции для Eigen::Matrix; для верификации результатов применять метод Eigen::MatrixXf::isApprox с абсолютной точностью 10−2.

4. Используя фреймворк Google Benchmark, разработать бенчмарки для operator* со следующими размерами матриц: n×n, где n ∈ {16,32,64,128, 256,512,1024}. Бенчмарки должны игнорировать время, затраченное на вы деление, копирование и освобождение памяти. Для корректного измерения времени выполнения CUDA-кода необходимо использовать CUDA Events API.

5. Построить график реальной вычислительной сложности умножения мат риц типа Matrix с помощью новой реализации operator*, а также анало гичный график для предыдущей версии operator*.

6. Построить график ускорения (speedup) новой реализации operator* отно сительно предыдущей версии.

7. Объяснить экспериментальные результаты.

8. Подготовить отчёт, содержащий:

- ключевые фрагменты реализованного кода;
- ссылку на репозиторий с полной реализацией;
- графики результатов измерений;
- анализ и интерпретацию полученных результатов.

Ключевые фрагменты кода

Ключевые моменты кода:

- Реализация matrixMultiplyKernelWMMA (исходный код приведен в приложении А);

Результат выполнения программы

На рисунках 1 и 2 представлены графики, построенные на основании выполненной программы.

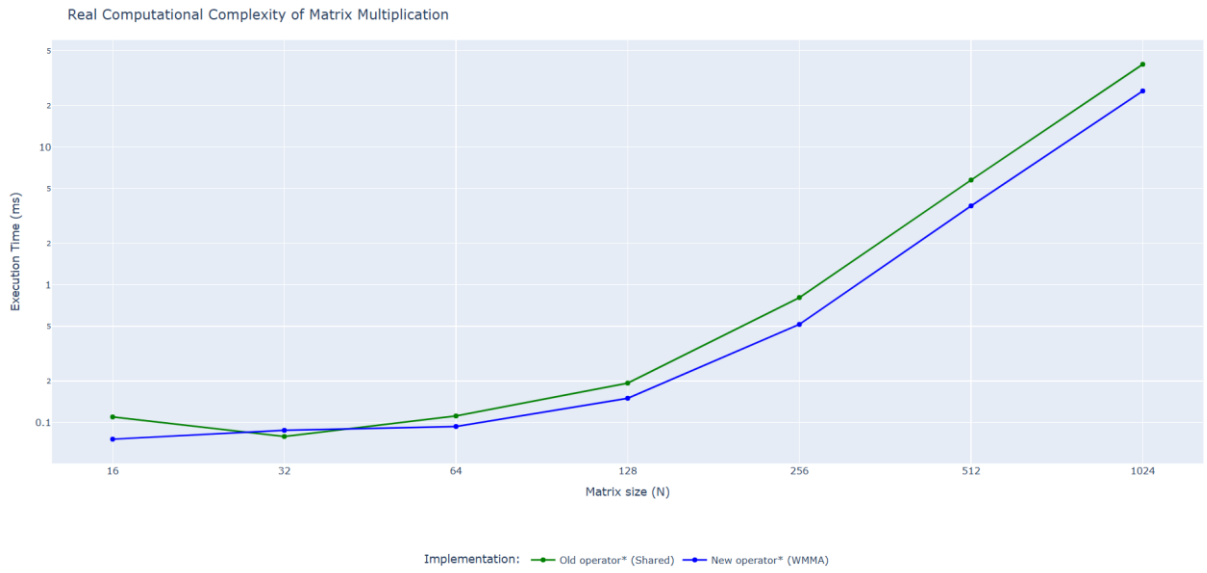

Рисунок 1 – График реальной вычислительной сложности

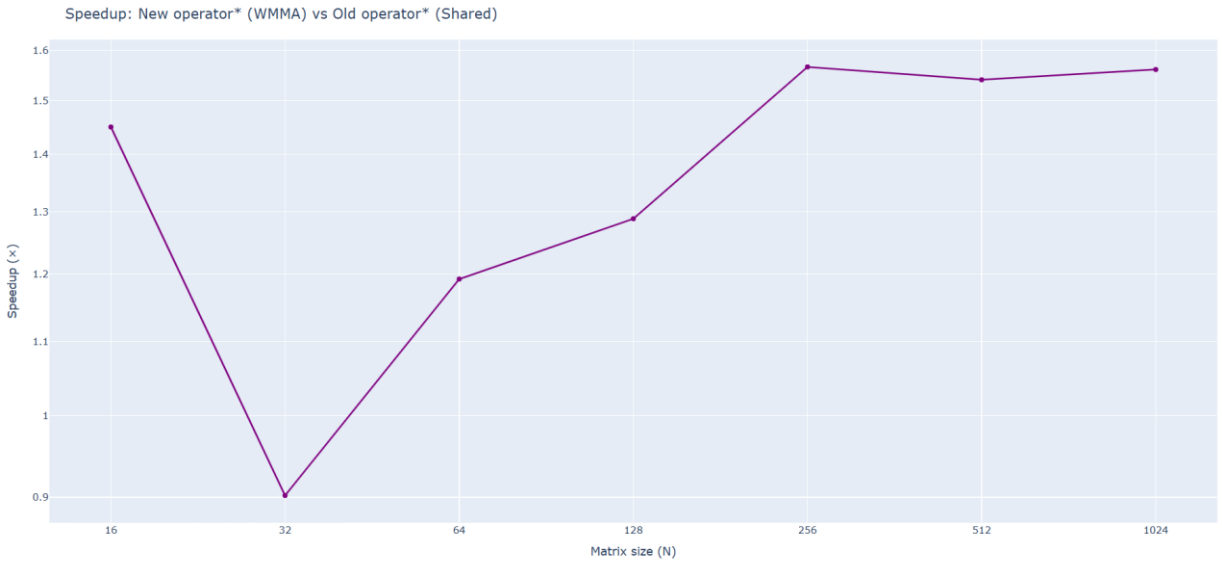

Рисунок 2 – График ускорения

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы были получены навыки программирования тензорных ядер CUDA.

ПРИЛОЖЕНИЕ А

```
68      template<typename T>
69      __global__ void matrixMultiplyKernelWMMA(MatrixView<T> A, MatrixView<T> B, MatrixView<T> C) {
70          using namespace nvcuda;
71
72
73          constexpr size_t wmma_m = 16;
74          constexpr size_t wmma_n = 16;
75          constexpr size_t wmma_k = type_id<T>();
76          constexpr size_t warpSize = 32;
77
78          const size_t warp_id = threadIdx.x / warpSize;
79
80          const size_t warpM = blockIdx.y * (blockDim.x / warpSize) + warp_id;
81          const size_t warpN = blockIdx.x;
82
83          if (warpM >= (C.rows() + wmma_m - 1) / wmma_m ||
84              warpN >= (C.cols() + wmma_n - 1) / wmma_n) {
85              return;
86          }
87
88
89          wmma::fragment<wmma::matrix_a, wmma_m, wmma_n, wmma_k, T, wmma::row_major> a_frag;
90          wmma::fragment<wmma::matrix_b, wmma_m, wmma_n, wmma_k, T, wmma::col_major> b_frag;
91          wmma::fragment<wmma::accumulator, wmma_m, wmma_n, wmma_k, T> c_frag;
92
93          wmma::fill_fragment(c_frag, 0.0f);
```

Рисунок А.1 – Реализация matrixMultiplyKernelWMMA

```
    if (warpM >= C.rows() / wmma_m || warpN >= C.cols() / wmma_n){
        return;
    }

    for (size_t i = 0; i < A.cols(); i += wmma_k) {
        if (i + wmma_k <= A.cols()) {
            wmma::load_matrix_sync(a_frag, &A(warpM * wmma_m, i), A.cols());
            wmma::load_matrix_sync(b_frag, &B(i, warpN * wmma_n), B.cols());

            wmma::mma_sync(c_frag, a_frag, b_frag, c_frag);
        }
    }

    wmma::store_matrix_sync(&C(warpM * wmma_m, warpN * wmma_n), c_frag, C.cols(), wmma::mem_row_major);


}
```

Рисунок А.2 – Реализация matrixMultiplyKernelWMMA