Практическая работа №3

ст. преп. каф. ВпВ ИКИТ СФУ Тарасов С. А.

Цель работы

Закрепить навыки работы с двумерными сетками нитей CUDA. Освоить работу с разделяемой памятью и примитивами синхронизации нитей.

Задание

1. Разработать кёрнел `kernel_matmul_shmem`, который принимает объекты `MatrixView` по значению и вычисляет произведение матриц, используя разделяемую память в качестве буфера для обмена данными между нитями (см. рис. 1). Для синхронизации нитей блока использовать `__syncthreads`; не использовать `CUDA Cooperative Groups`.

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 $(m \times k)$ и B $(k \times n)$, где $m, n, k \in \{1, 2, 3, 127, 128, 129, 512\}$. В качестве эталона для сравнения использовать результат аналогичной операции для `Eigen::MatrixXf`; для верификации результатов применять метод `Eigen::MatrixXf::isApprox` с абсолютной точностью 10^{-5}.

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

5. Построить график реальной вычислительной сложности умножения матриц типа `Matrix` с помощью новой реализации `operator*`, а также аналогичный график для предыдущей версии `operator*`.

6. Построить график ускорения (`speedup`) новой реализации `operator*` относительно предыдущей версии.

7. Сравнить экспериментальные результаты с теоретическими оценками вычислительной сложности.

8. Подготовить отчёт, содержащий:

   - ключевые фрагменты кода;

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

Критерии оценки

- **Корректность реализации и тестирование (50%):**
  - отсутствие утечек памяти, корректная работа с `CUDA API`;
  - правильность результатов умножения матриц различных размеров;
  - полнота тестового покрытия, включая граничные случаи;
  - соответствие результатов эталонной реализации.
- **Качество кода и архитектура (25%):**
  - чистота архитектуры, разделение ответственности между классами;
  - единообразие стиля, качество форматирования и читаемость кода.
- **Качество вычислительного эксперимента (15%):**
  - корректность методики измерений производительности;
  - глубина анализа результатов, сравнение с теоретическими оценками.
- **Документация и оформление (10%):**
  - полнота и структурированность отчёта;
  - ясность изложения;
  - оформление репозитория;
  - оформление отчёта (СТУ 7.5–07–2021).

Рекомендации по выполнению

- Используйте паттерн проектирования `стратегия` при разработке `operator*`.
- Ознакомьтесь с официальным руководством CUDA.

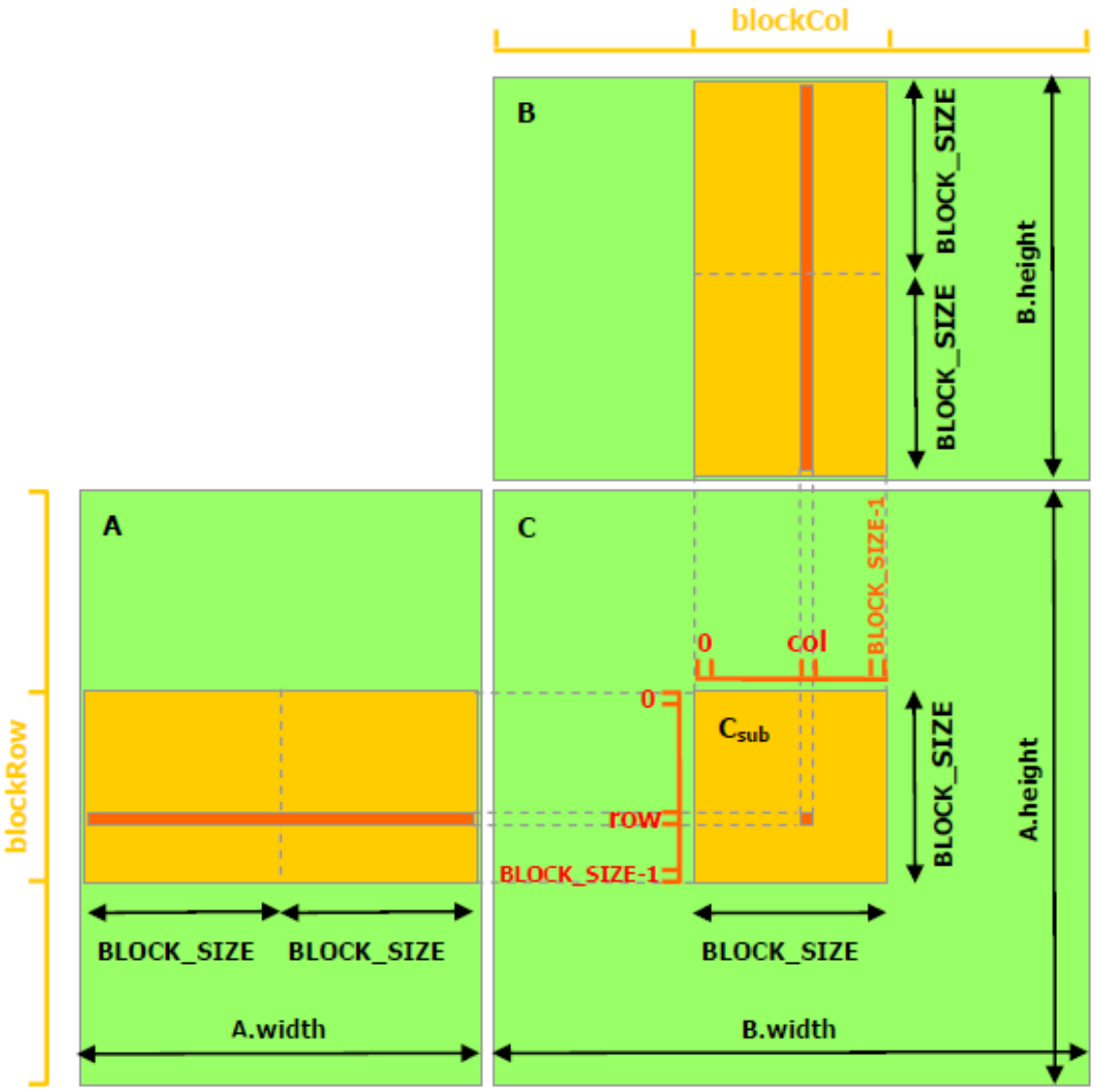

Рис. 1: Схема блочного алгоритма умножения матриц