Лабораторная работа №1

Максимов Артем ИВТ-21М

Задание:

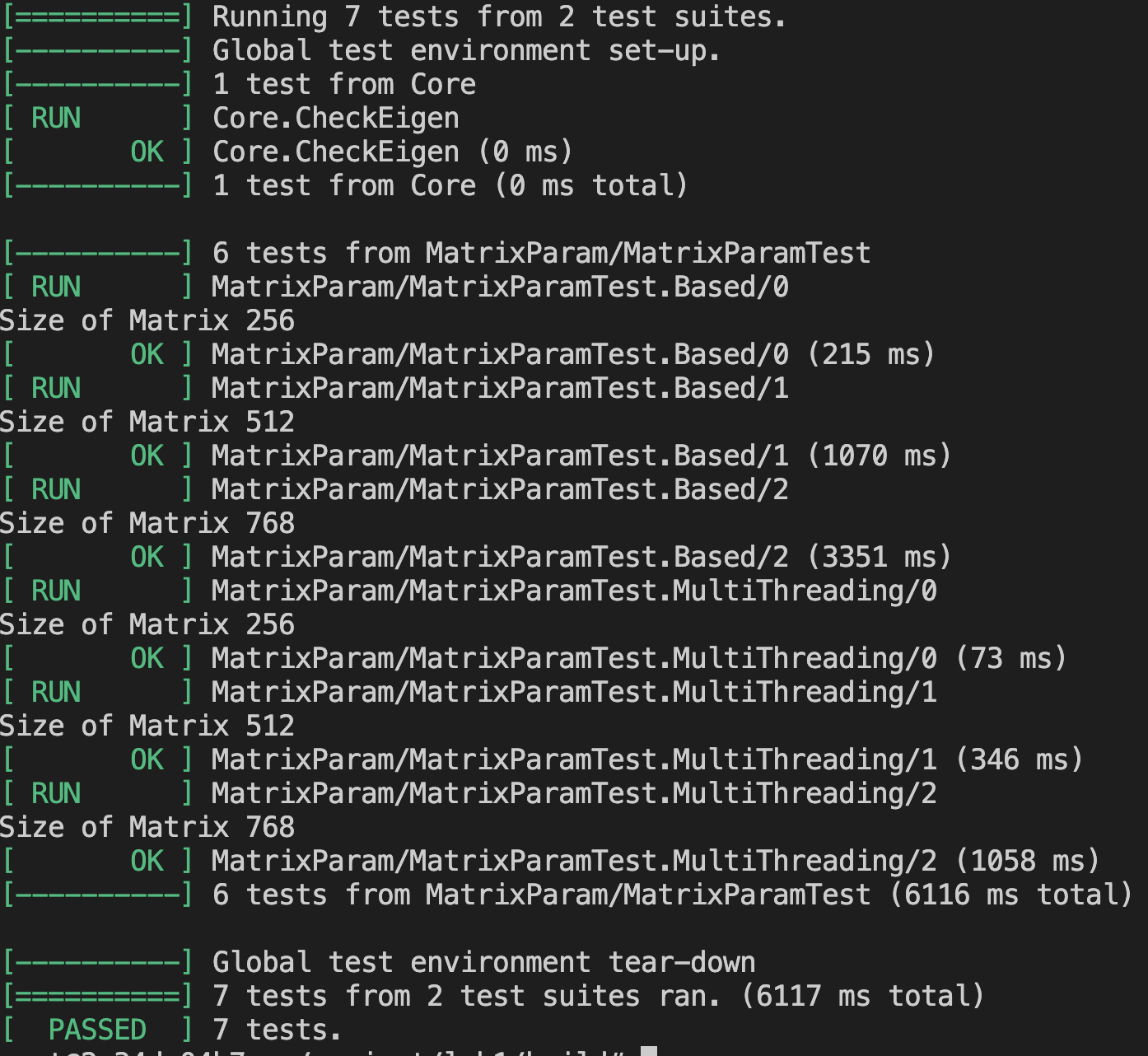
В работе необходимо составить и проверить алгоритм перемножения матриц. Для проверки алгоритма целесообразно использовать небольшие матрицы (до 3х3 включительно), результат перемножения вывести на экран. После успешной проверки задаются большие (1000х1000) матрицы как входные данные, наполнение матриц случайно и необязательно к выводу на экран. На скриншотах выполнения программы должна быть видна разница во времени исполнения в последовательном и в параллельном режимах

Решение

В процессе работы были использованы технологии google benchmark и google test для проверки работоспособности и производительно разработаных оптимизаций

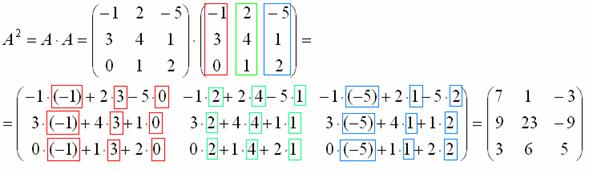
Также была использована библиотека Eigen (Eigen/Dense) для проверки корректного результата перемножения обоих реализованных вариантов перемножения матриц. Исходный код тестов находится в *labs/lab1/include/test.hpp*

Результаты тестирования



Исходный код обычного метода хранится в *labs/lab1/include/matrix\_mull.hpp* функция *matrix\_mull*

Основная идея реализовать такой алгоритм



Модернизированный вариант находится в *labs/lab1/include/matrix\_mull.hpp* функция *matrix\_mull\_perf*

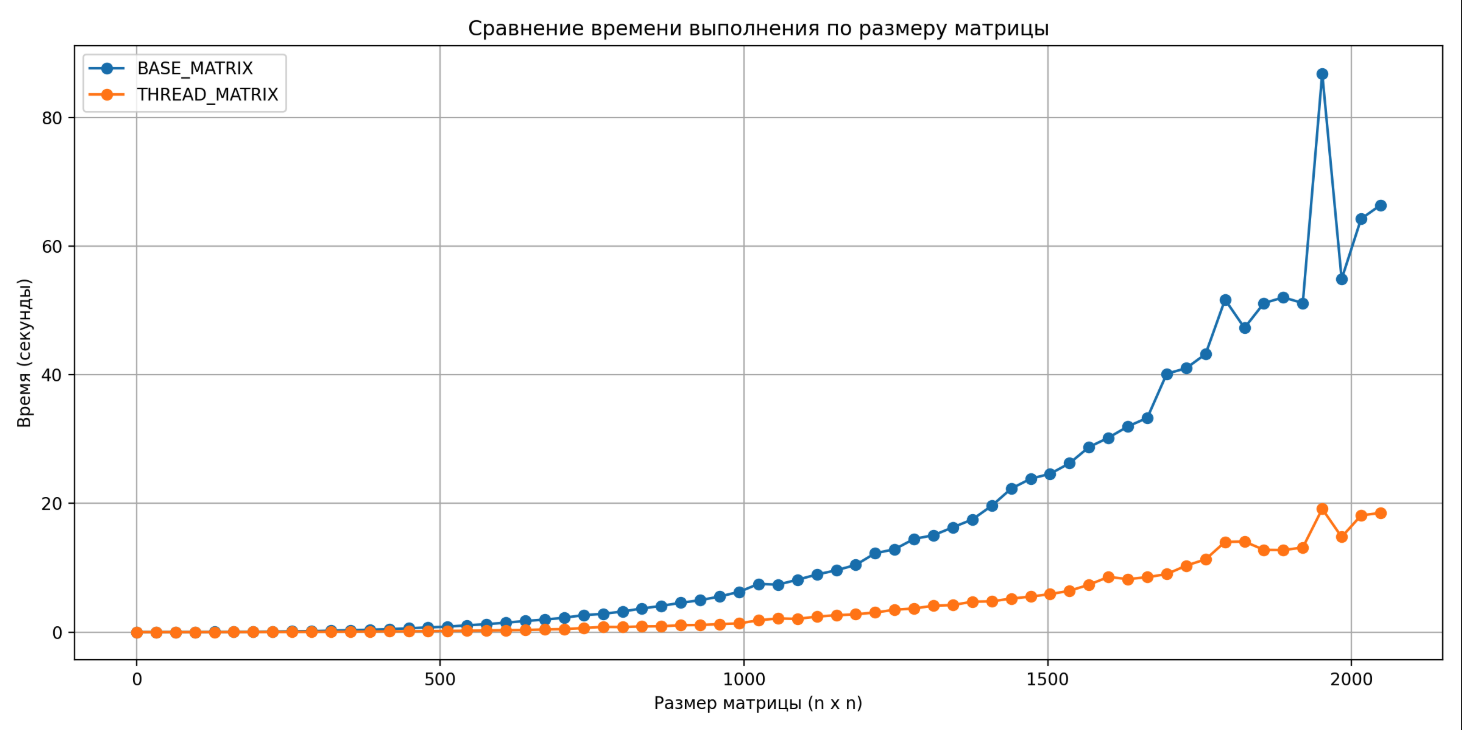
Основная идея распараллелить вычисление по потокам, где вычисляться в потоке будет по строкам

Использование не *scheduler(dynamic)* распределение задач при распараллеливании связано с тем, что вычисление для каждого строки по сути своей является примерно одинаковой по загруженности задачей, так что добавлять динамическое распределение не имеет смысла

Также появилась переменная *sum,* которая является по своей сути хранилищем для текущего значения и позволяет не обращаться постоянно к элементу матрицы, что делает количество “далеких” обращения к участку памяти намного меньше

Результат приведен в stats/img в виде графика сравнения real\_time времени выполнения двух методов на разных данных

Итоговые данные



Как видно, что при размерах матрицы 1000x1000 разница имеется и многопоточности работает быстро, однако не настолько быстрей т.к время показано на размере 2000x2000, что говорит о том, что задачи по вычислению строки сами по себе довольно легкие, однако с увеличением количеством этих строк и столбцом, на которые происходит перемножение, не учитывая cache-miss’s, многопоточности показывает отличные результаты.