Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский химико-технологический университет имени Д.И. Менделеева»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Выполнил студент группы КС-36: Золотухин А.А.
Ссылка на репозиторий: https://github.com/CorgiPuppy/parallel-prog-labs
Принял: Бабкин Михаил Андреевич
Дата сдачи: 01.10.2025

Москва
2025

Оглавление

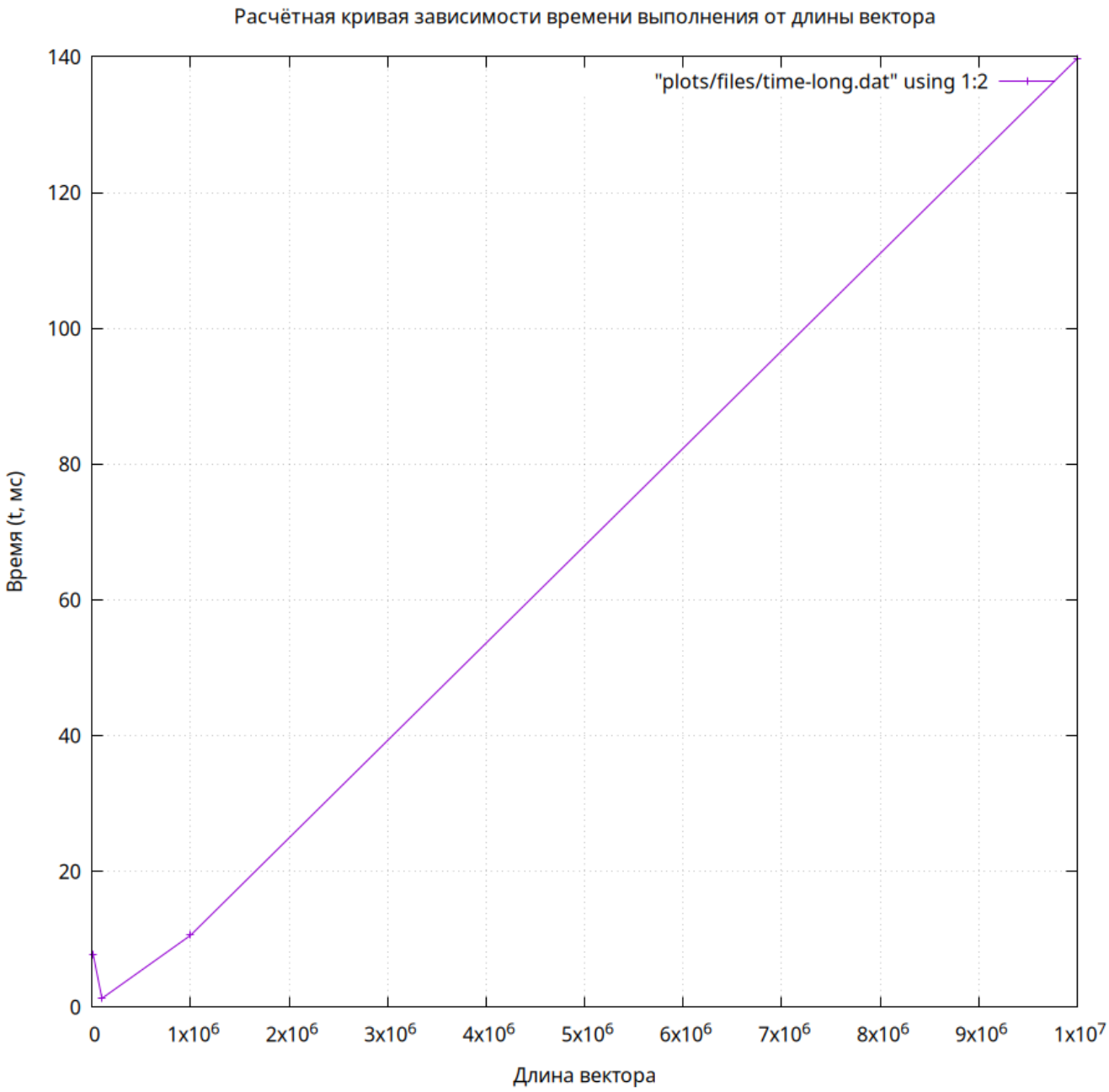
Описание задачи

Найти скалярное произведение векторов A и B, заполненных случайными целыми числами от 0 до 100. Длина векторов выбирается из множества 10000; 100000; 1000000; 10000000 (рассмотреть 4 случая). Использовать 10 потоков. Измерить время работы программы в каждом случае, построить график зависимости времени выполнения от длины вектора.

Выполнение задачи

```cpp
#include <iostream>
#include <vector>
#include <random>
#include <algorithm>
#include <chrono>
#include <fstream>
#include <thread>

#include "../include/Constants.h"

void generateVector(std::vector<int> &);
void calculateProductParallel(std::vector<int>, std::vector<int>);
void calculateScalarProduct(const std::vector<int>&, const std::vector<int>&, int, int, int&);

int main() {
    std::ofstream time_long(Constants::folder + "time-long.dat");

    for (int i = 0; i < (int)Constants::longOfVector.size(); i++) {
        std::vector<int> vectorA(Constants::longOfVector[i]);
        std::vector<int> vectorB(Constants::longOfVector[i]);

        generateVector(vectorA);
        generateVector(vectorB);

        std::chrono::high_resolution_clock::time_point start = std::chrono::high_resolution_clock::now();

        calculateProductParallel(vectorA, vectorB);

        std::chrono::high_resolution_clock::time_point end = std::chrono::high_resolution_clock::now();
        std::chrono::duration<long double, std::milli> milli_diff = end - start;
        long double time_taken = milli_diff.count();

        time_long << Constants::longOfVector[i] << " " << time_taken << std::endl;
    }

    time_long.close();

    return 0;
}

void generateVector(std::vector<int> &vector) {
```

```
    std::random_device rnd_device;
    std::mt19937 mersenne_engine {rnd_device()};
    std::uniform_int_distribution<int> dist {Constants::minNumberOfVector, Constants
    ::maxNumberOfVector};

    auto gen = [&]() {
        return dist(mersenne_engine);
    };

    std::generate(vector.begin(), vector.end(), gen);
}

void calculateProductParallel(std::vector<int> firstVector, std::vector<int>
    secondVector) {
    int total_scalar_product = 0;

    std::vector<std::thread> threads;
    std::vector<int> partial_results(Constants::numberOfThreads, 0);

    for (int i = 0; i < Constants::numberOfThreads; i++) {
        int start = i * (firstVector.size() / Constants::numberOfThreads);

    int end;
    if (i == Constants::numberOfThreads - 1)
      end = firstVector.size();
    else
      end = (i + 1) * (firstVector.size() / Constants::numberOfThreads);

        threads.emplace_back(calculateScalarProduct, std::cref(firstVector), std::
    cref(secondVector), start, end, std::ref(partial_results[i]));
    }

    for (auto& thread : threads) {
        thread.join();
    }

    for (int i = 0; i < Constants::numberOfThreads; i++) {
        total_scalar_product += partial_results[i];
    }
}

void calculateScalarProduct(const std::vector<int>& firstVector, const std::vector<
    int>& secondVector, int start, int end, int& result) {
    result = 0;
    for (int i = start; i < end; i++) {
        result += firstVector[i] * secondVector[i];
    }
}
```


Расчётная кривая зависимости времени выполнения от длины вектора
"plots/files/time-long.dat" using 1:2
Время (t, мс)
Длина вектора
0
20
40
60
80
100
120
140
0
1x10^6
2x10^6
3x10^6
4x10^6
5x10^6
6x10^6
7x10^6
8x10^6
9x10^6
1x10^7