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 1
Выполнение задачи 1

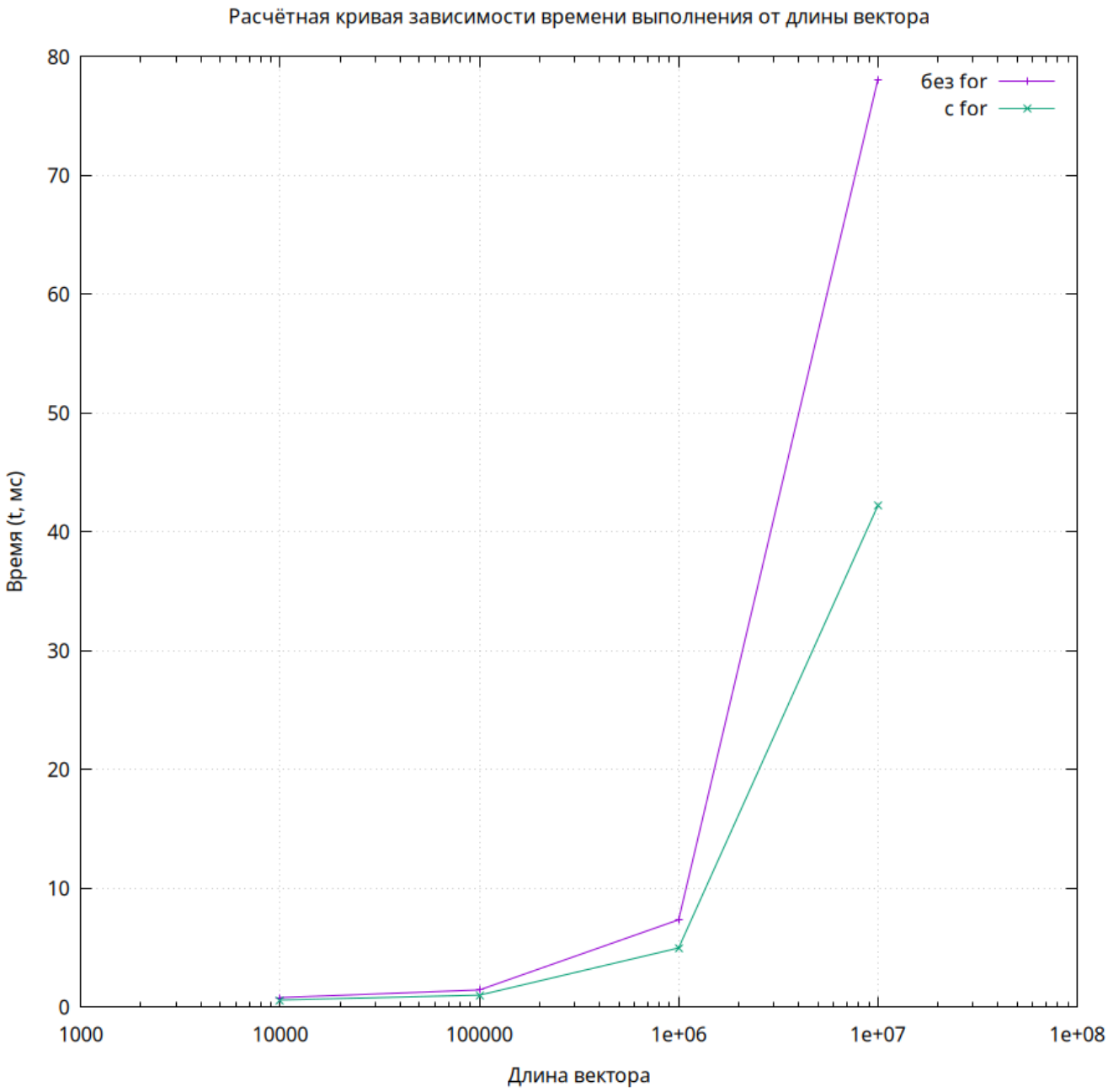
Описание задачи

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 <omp.h>

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

void generateVector(std::vector<int> &);
void calculateProductParallelNoFor(std::vector<int>, std::vector<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();
        calculateProductParallelNoFor(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_taken1 = milli_diff.count();

        start = std::chrono::high_resolution_clock::now();
        calculateProductParallelNoFor(vectorA, vectorB);
        end = std::chrono::high_resolution_clock::now();
        milli_diff = end - start;
        long double time_taken2 = milli_diff.count();

        time_long << Constants::longOfVector[i] << " " << time_taken1 << " " << time_taken2 << 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 calculateProductParallelNoFor(std::vector<int> firstVector, std::vector<int>
    secondVector) {
    int total_scalar_product = 0;
    int size = firstVector.size();

  #pragma omp parallel num_threads(Constants::numberOfThreads)
  {
    int thread_id = omp_get_thread_num();
    int num_threads = omp_get_num_threads();

    int start = thread_id * (size / num_threads);

    int end;
    if (thread_id == num_threads - 1)
      end = size;
    else
      end = (thread_id + 1) * (size / num_threads);

    int partial_result = 0;
    for (int i = start; i < end; i++)
      partial_result += firstVector[i] * secondVector[i];

    #pragma omp critical
        total_scalar_product += partial_result;
  }
}

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

  #pragma omp parallel num_threads(Constants::numberOfThreads) reduction(+:
    total_scalar_product)
  {
    #pragma omp for
    for (int i = 0; i < size; i++)
      total_scalar_product += firstVector[i] * secondVector[i];
  }
}
```


Расчётная кривая зависимости времени выполнения от длины вектора
без for
с for
Время (t, мс)
Длина вектора
0
10
20
30
40
50
60
70
80
1000
10000
100000
1e+06
1e+07
1e+08