Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.04 – «Программная инженерия»

Дисциплина: «Технологии блокчейн и распределенные информационные системы»

Семестр 8

ОТЧЕТ

по лабораторной работе №6

Тема: «Решение СЛАУ с помощью библиотеки OneAPI Threading Building Blocks»

Выполнил: студент группы РИС-20-2б

Гулуа И.Ю. \_\_\_\_\_\_\_\_\_\_\_

Проверил: старший преподаватель кафедры ИТАС

Щапов В. А. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2024

**Цели работы**

Распараллеливание метода Гаусса с помощью библиотеки OneAPI Threading Building Blocks (oneTBB) для решения систем линейных алгебраических уравнений.

**Задание к работе**

1. Разработать параллельный алгоритм, написать и отладить параллельную программу решения СЛАУ методом Гаусса в oneTBB.

**Ход работы**

Напишем на языке программирования C++ программу для решения СЛАУ с помощью метода Гаусса, с применением oneTBB. Метод Гаусса представлена на листинге 1:

Листинг 1 – Код программы

#include <tbb/tbb.h>

#include <chrono>

#include <iostream>

using namespace std;

int Rrand(int min, int max) {

return 1 + rand() % (max - min + 1) + min;

}

double\* GaussTBB(double\*\* A, double\* Y, int n) {

double\* X = new double[n];

for (int k = 0; k < n; k++)

{

tbb::parallel\_for(tbb::blocked\_range<int>(k + 1, n), [&](const tbb::blocked\_range<int>& range) {

for (int j = range.begin(); j < range.end(); j++) {

double m = A[j][k] / A[k][k];

for (int i = k; i < n; i++) {

A[j][i] = A[j][i] - m \* A[k][i];

}

Y[j] = Y[j] - m \* Y[k];

}

});

}

for (int k = n - 1; k >= 0; k--) {

X[k] = Y[k];

for (int i = k + 1; i < n; i++) {

X[k] = X[k] - A[k][i] \* X[i];

}

X[k] = X[k] / A[k][k];

}

return X;

}

double\* FindY(double\*\* A, double\* X, int n) {

double\* Y = new double[n];

for (int i = 0; i < n; i++) {

Y[i] = 0;

for (int j = 0; j < n; j++) {

Y[i] += A[i][j] \* X[j];

}

}

return Y;

}

double\* Gauss(double\*\* A, double\* Y, int n) {

double\* X = new double[n];

for (int k = 0; k < n; k++)

{

for (int j = k + 1; j < n; j++) {

double m = A[j][k] / A[k][k];

for (int i = k; i < n; i++) {

A[j][i] = A[j][i] - m \* A[k][i];

}

Y[j] = Y[j] - m \* Y[k];

}

}

for (int k = n - 1; k >= 0; k--) {

X[k] = Y[k];

for (int i = k + 1; i < n; i++) {

X[k] = X[k] - A[k][i] \* X[i];

}

X[k] = X[k] / A[k][k];

}

return X;

}

int ArrEquial(double\* X1, double\* X2, int n, double eps) {

int c = 0;

for (int i = 0; i < n; i++) {

if (std::abs(X1[i] - X2[i]) > eps) {

c++;

}

}

return c;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int nX=3200;

int f = 0;

double\*\* A = new double\* [nX];

for (int i = 0; i < nX; i++) {

A[i] = new double[nX];

}

for (int i = 0; i < nX; i++) {

for (int j = 0; j < nX; j++) {

A[i][j] = 0;

}

}

double\* X = new double[nX];

double\* X2 = new double[nX];

double\* Y = new double[nX];

for (int i = 0; i < nX; i++) { //заполнение матрицы

for (int j = 0; j < nX; j++) {

A[i][j] = Rrand(-100, 100);

}

}

for (int i = 0; i < nX; i++) { //заполнение правой части

X2[i] = Rrand(-100, 100);

}

Y = FindY(A, X2, nX);

std::cout << "OneTBB решение" << std::endl;

auto begin = std::chrono::steady\_clock::now();

X = GaussTBB(A, Y, nX);

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << "Время: " << elapsed\_ms.count() << " ms\n";

f = ArrEquial(X, X2, nX, 0.001);

std::cout << "Количество несовпадений:" << f << std::endl;

std::cout << "Прямое решение" << std::endl;

begin = std::chrono::steady\_clock::now();

X = Gauss(A, Y, nX);

end = std::chrono::steady\_clock::now();

elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << "Время: " << elapsed\_ms.count() << " ms\n";

f = ArrEquial(X, X2, nX, 0.001);

std::cout << "Количество несовпадений:" << f << std::endl;

for (int i = 0; i < nX; i++) {

delete[] A[i];

}

delete[] A;

delete[] X;

delete[] Y;

}

Метод Гаусса состоит из прямого и обратного хода. Распараллеливание прямого хода с помощью функции parallel\_for() значительно ускоряет процесс. Программа автоматически генерирует матрицу весов и вектор неизвестных, а затем вычисляет вектор свободных членов для проверки решения. После обратного хода полученный вектор неизвестных сравнивается с исходным. Если все значения совпадают, метод сработал верно.

Результат работы программы представлен на рисунке 1.

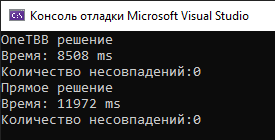


Рисунок 1 – Выполнение программы

На рисунке 1 показано, что для системы с 3200 неизвестными параллельная версия метода Гаусса завершилась за 8,5 секунд, а непараллельная - за 11,9 секунд.

**Заключение**

В ходе выполнения лабораторной работы была написана программа для решения системы линейных алгебраических уравнений с помощью метода Гаусса. Применена библиотека oneTBB для распараллеливания метода Гаусса. Цели работы достигнуты, все задачи выполнены.