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 подготовки: 09.03.01 Информатика и вычислительная техника

Дисциплина: «Технологии блокчейн и распределенные информационные системы»

**ОТЧЕТ**

по лабораторной работе №3

Выполнил: студент группы АСУ-19-1б

Нерослов Е. А. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Щапов В. А. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2023

ЗАДАНИЕ

Реализовать умножение матриц последовательно и с помощью OpenMP. Учесть архитектуру кэширования при выборе алгоритма и стратегии распараллеливания.

КОД

#include <iostream>

#include <cstdlib>

#include <omp.h>

using namespace std;

int main()

{

system("chcp 1251");

system("cls");

srand(time(NULL));

int matrix\_size;

cout << "Введите размер матриц: ";

cin >> matrix\_size;

int row1 = matrix\_size, row2 = matrix\_size, col1 = matrix\_size, col2 = matrix\_size;

int\*\* a, \*\* b, \*\* c;

// Ввод элементов первой матрицы

a = new int\* [row1];

for (int i = 0; i < row1; i++)

{

a[i] = new int[col1];

for (int j = 0; j < col1; j++)

{

a[i][j] = rand() % 100 + 1;

}

}

// Вывод элементов первой матрицы

//for (int i = 0; i < row1; i++)

//{

// for (int j = 0; j < col1; j++)

// cout << a[i][j] << "\t";

// cout << endl;

//}

// Ввод элементов второй матрицы

b = new int\* [row2];

for (int i = 0; i < row2; i++)

{

b[i] = new int[col2];

for (int j = 0; j < col2; j++)

{

b[i][j] = rand() % 100 + 1;

}

}

// Вывод элементов второй матрицы

//for (int i = 0; i < row2; i++)

//{

// for (int j = 0; j < col2; j++)

// {

// cout << b[i][j] << "\t";

// }

// cout << endl;

//}

// Умножение матриц

c = new int\* [row1];

for (int i = 0; i < row1; i++)

{

c[i] = new int[col2];

for (int j = 0; j < col2; j++)

{

c[i][j] = 0;

}

}

double seconds = clock();

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col2; j++)

{

for (int k = 0; k < col1; k++)

c[i][j] += a[i][k] \* b[k][j];

}

}

seconds = (clock() - seconds) / 1000;

printf("Время без ускорением: %f seconds\n", seconds);

// Умножение матриц

c = new int\* [row1];

for (int i = 0; i < row1; i++)

{

c[i] = new int[col2];

for (int j = 0; j < col2; j++)

{

c[i][j] = 0;

}

}

seconds = clock();

for (int i = 0; i < row1; i++)

{

for (int k = 0; k < col1; k++)

{

for (int j = 0; j < col2; j++)

c[i][j] += a[i][k] \* b[k][j];

}

}

seconds = (clock() - seconds) / 1000;

printf("Время с ускорением: %f seconds\n", seconds);

// Умножение матриц

c = new int\* [row1];

for (int i = 0; i < row1; i++)

{

c[i] = new int[col2];

for (int j = 0; j < col2; j++)

{

c[i][j] = 0;

}

}

seconds = clock();

int i, j, k;

#pragma omp parallel for shared(a, b, c) private(i, j, k)

for (i = 0; i < row1; i++)

{

for (k = 0; k < col1; k++)

{

for (j = 0; j < col2; j++)

c[i][j] += a[i][k] \* b[k][j];

}

}

seconds = (clock() - seconds) / 1000;

printf("Время с ускорением и с распараллеливанием: %f с.\n", seconds);

// Вывод матрицы произведения

//cout << "Матрица произведения" << endl;

//for (int i = 0; i < row1; i++)

//{

// for (int j = 0; j < col2; j++)

// cout << c[i][j] << "\t";

// cout << endl;

//}

ТЕСТЫ

|  |  |  |  |
| --- | --- | --- | --- |
| Размер | Время без ускорения, с | Время с ускорением, с | Время с ускорением и с распараллеливанием, с |
| 500 | 0,56 | 0,477 | 0,118 |
| 1000 | 7,266 | 3,908 | 0,906 |
| 1500 | 33,435 | 13,348 | 3,05 |
| 2000 | 90,746 | 32,08 | 7,736 |
| 2500 | 173,399 | 62,412 | 18,155 |
| 3000 | 372,758 | 107,681 | 32,421 |

|  |  |  |  |
| --- | --- | --- | --- |
| Размер | Время без ускорения, Размер/с | Время с ускорением, Размер/с | Время с ускорением и с распараллеливанием, Размер/с |
| 500 | 892,8571429 | 1048,218029 | 4237,288136 |
| 1000 | 137,6273053 | 255,8853634 | 1103,752759 |
| 1500 | 44,86316734 | 112,376386 | 491,8032787 |
| 2000 | 22,03953893 | 62,34413965 | 258,5315408 |
| 2500 | 14,41761487 | 40,05639941 | 137,7031121 |
| 3000 | 8,048117009 | 27,86006816 | 92,53261775 |