Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Теория распределенных систем и параллельных вычислений”

Лабораторная работа №1

“Исследование средств создания распределено выполняющихся программ”

Выполнил:

ст.гр. ИС/б-17-2

Долженко И.А.

Проверил:

Дрозин А.Ю.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Исследовать функции библиотеки MPI, необходимые для создания и взаимодействия распределено выполняемых программ.

2 ПОСТАНОВКА ЗАДАЧИ

**Вариант №2.** Программа осуществляет вычисление определителя матрицы 4\*4 методом треугольников. Каждый процесс подсчитывает только произведения, определение результата осуществляется в родительской задаче, куда передаются результаты работы процесса. По процессам распределяется вся матрица.

3 КОД ПРОГРАММЫ

#include <iostream>

#include <fstream>

#include <vector>

#include "mpi.h"

using namespace std;

vector<vector<int>> readMatrixFromFile(string path, int\* row, int\* column);

vector<vector<int>> getSubMatrix(vector<vector<int>>, int, int);

void printMatrix(vector<vector<int>>);

int\* matrixToVector(vector<vector<int>>);

int getMinor(int v[9]);

int row, column;

int determinant;

int main(int argc, char\* argv[])

{

MPI\_Status status;

int rank, size, i = 0;

int minorSize;

vector< vector<int> > matrix, minorMatrix;

int\* minorVector;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

matrix = readMatrixFromFile("matrix.txt", &row, &column);

minorSize = (matrix.size() - 1) \* (matrix[0].size() - 1);

if (rank == 0) {

cout << "-------------------------------" << endl;

cout << "Matrix:" << endl;

printMatrix(matrix);

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

minorVector = matrixToVector(getSubMatrix(matrix, i, j));

MPI\_Ssend(minorVector, minorSize, MPI\_INT, (j + 1), 0, MPI\_COMM\_WORLD);

}

int result;

for (int j = 1; j < size; j++) {

MPI\_Recv(&result, 1, MPI\_INT, j, 0, MPI\_COMM\_WORLD, &status);

determinant += result;

}

}

else {

int\* vector = new int[minorSize];

MPI\_Recv(vector, minorSize, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, &status);

int answer = matrix[i][rank - 1] \* pow((-1), (1 + rank)) \* getMinor(vector);

MPI\_Send(&answer, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

MPI\_Barrier(MPI\_COMM\_WORLD);

MPI\_Finalize();

if (rank == 0) {

cout << "Determinant: " << determinant << endl;

cout << "-------------------------------" << endl;

}

return 0;

}

void printMatrix(vector<vector<int>> matrix) {

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix[0].size(); j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

}

vector<vector<int>> getSubMatrix(vector<vector<int>> matrix, int rowErasePosition, int columnErasePosition) {

matrix.erase(matrix.begin() + rowErasePosition);

for (int i = 0; i < matrix.size(); i++) {

matrix[i].erase(matrix[i].begin() + columnErasePosition);

}

return matrix;

}

int getMinor(int v[9]) {

int temp = v[0] \* v[4] \* v[8] + v[1] \* v[5] \* v[6] + v[2] \* v[3] \* v[7] - v[0] \* v[5] \* v[7] - v[1] \* v[3] \* v[8] - v[2] \* v[4] \* v[6];

return temp;

}

int\* matrixToVector(vector<vector<int>> matrix) {

int row = matrix.size(), column = matrix[0].size();

int\* result = new int[row \* column];

int t = 0;

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

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

result[t++] = matrix[i][j];

}

}

return result;

}

vector<vector<int>> readMatrixFromFile(string path, int\* row, int\* column) {

ifstream in(path);

in >> \*row >> \*column;

vector<vector <int> > matrix(\*row);

for (int i = 0; i < \*row; i++) {

matrix[i].resize(\*column);

}

for (int i = 0; i < \*row; i++) {

for (int j = 0; j < \*column; j++) {

in >> matrix[i][j];

}

}

in.close();

return matrix;

}

4 РЕЗУЛЬТАТЫ

Заполним файл matrix.txt такой матрицей 4x4:

4 4

0 -4 1 0

5 9 0 3

0 -2 8 4

8 9 13 17

Далее запустим с помощью команды mpiexec -n 5 .\ConsoleApplication1.exe

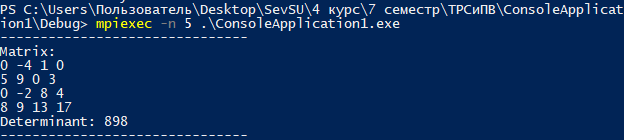


Рисунок 1 – Результат выполнения программы

Проверим результат с помощью онлайн калькулятора.

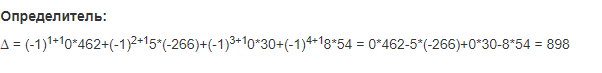


Рисунок 2 – Проверка результата

ВЫВОД

В ходе выполнения лабораторной работы были исследованы функции библиотеки MPI, необходимые для создания и взаимодействия распределено выполняемых программ. Создана программа для вычисления определителя матрицы 4\*4 методом треугольников. Каждый процесс подсчитывает только произведения, определение результата осуществляется в родительской задаче, куда передаются результаты работы процесса. По процессам распределяется вся матрица.