Міністерство освіти і науки України

Одеський національний політехнічний університет

Інститут комп’ютерних систем  
Кафедра комп’ютерних інтелектуальних систем і мереж

Протокол

Лабораторної роботи №5

На тему “Разработка параллельного алгоритма умножение вектора на матрицу, ленточный алгоритм. Оценка эффективности параллельного алгоритма”

По предмету: «Основи розподілених обчислень»

Виконали:

студенти групи АМ-181

Батаев М.С.

Борщов М.С.

Карабань О.С.

Перевірив:

Тішин П.М.

Одеса 2021

**Перелік завдань до лабораторної роботи**

1. Напишите и отладьте программу умножения матрицы на вектор.

2. Элементы матрицы  и вектора  должны определяться по формулам:

,

, , 

где  - номер варианта,  - размерность матрицы.

**Хід роботи**

**Код програми**

#include <mpi.h>

#include <iostream>

#include <windows.h>

#include <math.h>

#include <ctime>

using namespace std;

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

{

srand(time(0));

const int MAX\_ROWS = 10;

const int MAX\_COLS = 10;

int rows, cols;

double A[MAX\_COLS][MAX\_ROWS], B[MAX\_COLS], C[MAX\_ROWS], buffer[MAX\_COLS], ans;

int myid, numprocs;

int i, j, anstype, row;

MPI\_Status status;

MPI\_Init(&argc, &argv);

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

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numprocs); //определяем кол-во процессов

rows = MAX\_ROWS;

cols = MAX\_COLS;

if (myid == 0) {

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

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

A[j][i] = double(rand() % 10) + 1; // Заполнение матрицы А случайными числами

}

B[j] = double(j+1) / cols; // Заполнение вектора B числами по формуле B=i/n

}

MPI\_Bcast(&B, cols, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD); // Рассылка сообщения от 0 процесса всем процессам, включая 0 процесс

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

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

buffer[i] = A[j][i];

}

if (j != 9) {

MPI\_Send(&buffer, cols, MPI\_DOUBLE, (j + 1), (j + 1), MPI\_COMM\_WORLD); // Рассылка элементов матрицы в процессы

}

else {

MPI\_Send(&buffer, cols, MPI\_DOUBLE, j, j, MPI\_COMM\_WORLD);

}

}

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

MPI\_Recv(&ans, 1, MPI\_DOUBLE, MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status); // Прием сообщений

cout << "Result: " << ans;

anstype = status.MPI\_TAG;

C[anstype] = ans;

}

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

cout << "Result #" << i << ": " << C[i] << endl;

}

MPI\_Finalize();

}

else {

MPI\_Bcast(&B, cols, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD); //Рассылка сообщений всем процессам от 0 процесса

MPI\_Recv(&buffer, cols, MPI\_DOUBLE, 0, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status); // Прием сообщений

if (status.MPI\_TAG != 0) {

row = status.MPI\_TAG;

ans = 0.0;

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

ans += buffer[i] \* B[i];

}

MPI\_Send(&ans, 1, MPI\_DOUBLE, 0, row, MPI\_COMM\_WORLD);

}

for (int b = 0; b < MAX\_COLS; b++) {

result[b] = 0;

}

for (int a = 0; a < MAX\_ROWS; a++) {

MPI\_Bcast(B[a], 10, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

}

MPI\_Recv(&buffer, MAX\_ROWS, MPI\_DOUBLE, 0, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status);

if (status.MPI\_TAG != 0) {

for (int a = 0; a < MAX\_ROWS; a++) {

result[a] += buffer[a] \* B[a];

}

cout << status.MPI\_TAG << ": ";

for (int a = 0; a < MAX\_ROWS; a++) {

cout << result[a] << " ";

}

cout << endl;

}

}

MPI\_Finalize();

return 0;

}

}

**Висновок:**

У результаті виконання даної лабораторної роботи ми розробили програму паралельного множення вектора на матрицю шляхом використання методів MPI та зміцнили свої знання у передачі даних між процесами за допомогою методів MPI\_Send, MPI\_Bcast, MPI\_Recv.