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»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

по дисциплине **«Технологии блокчейн и распределенные информационные системы»**

**Выполнил** студент гр. РИС-20-2б

Уржумов В.И.

**Проверил** доц. Щапов В.А.

Пермь, 2024 год

Рассмотреть параллельное умножение матриц с помощью MPI.

**Задание к работе**

1. Написать программу для параллельного умножения матриц с помощью MPI

**Ход работы**

Напишем на языке программирования C++ программу для параллельного умножения матриц с помощью MPI. Функция main представлена на листинге 1:

Листинг 1 – Функция main

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <mpi.h>

#define N 2500

double matrix1[N][N];

double matrix2[N][N];

double matrixResult[N][N];

void printMatrix(double matrix[N][N]) {

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

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

printf("%f ", matrix[i][j]);

}

printf("\n");

}

}

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

{

int numt, rank;

int rowsForA, extraRows, rows;

int buff = 0;

MPI\_Status status;

double beginTime, endTime;

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

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

matrixResult[i][j] = 0;

}

}

MPI\_Init(&argc, &argv);

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

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

MPI\_Barrier(MPI\_COMM\_WORLD);

if (rank == 0) {

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

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

matrix1[i][j] = rand();

}

}

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

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

matrix2[i][j] = rand();

}

}

beginTime = MPI\_Wtime();

rowsForA = N / (numt - 1);

extraRows = N % (numt - 1);

for (int i = 1; i <= numt - 1; i++) {

if (i <= extraRows) {

rows = rowsForA + 1;

}

else {

rows = rowsForA;

}

MPI\_Send(&buff, 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD);

MPI\_Send(&rows, 1, MPI\_INT, i, 1, MPI\_COMM\_WORLD);

MPI\_Send(&matrix1[buff][0], N \* N, MPI\_DOUBLE, i, 1, MPI\_COMM\_WORLD);

MPI\_Send(&matrix2, N \* N, MPI\_DOUBLE, i, 1, MPI\_COMM\_WORLD);

buff += rows;

}

for (int i = 1; i <= numt - 1; i++) {

MPI\_Recv(&buff, 1, MPI\_INT, i, 2, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&rows, 1, MPI\_INT, i, 2, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&(matrixResult[buff][0]), N \* N, MPI\_DOUBLE, i, 2, MPI\_COMM\_WORLD, &status);

}

endTime = MPI\_Wtime();

double time = endTime - beginTime;

printf("Time %f s\n", time);

}

else {

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

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

MPI\_Recv(&matrix1[buff][0], N \* N, MPI\_DOUBLE, 0, 1, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&matrix2, N \* N, MPI\_DOUBLE, 0, 1, MPI\_COMM\_WORLD, &status);

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

for (int k = 0; k < N; k++) {

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

matrixResult[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

MPI\_Send(&buff, 1, MPI\_INT, 0, 2, MPI\_COMM\_WORLD);

MPI\_Send(&rows, 1, MPI\_INT, 0, 2, MPI\_COMM\_WORLD);

MPI\_Send(&(matrixResult[buff][0]), N \* N, MPI\_DOUBLE, 0, 2, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

}

Процесс с рангом 0 принимает на себя роль мастера, который распределяет строки матрицы 1 между остальными процессами. В отправки сообщений используется метод MPI\_Send, для принятия сообщений MPI\_Recv.

Пример работы программы представлен на рисунке 1.

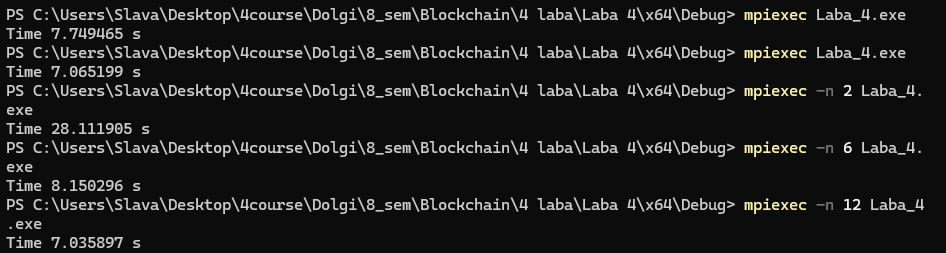


Рисунок 1 – Выполнение программы

На рисунке 1 представлен запуск программы для умножения матриц размерности 2500 на 2500

**Заключение**

В ходе выполнения лабораторной работы была написана программа программу для параллельного умножения матриц с помощью MPI.