Министерство науки и Высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №5
«Исследование моделей взаимодействия распределенно выполняющихся процессов»
по дисциплине
«ТЕОРИЯ РАСПРЕДЕЛЕННЫХ СИСТЕМ И ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ»

Выполнил студент группы ИС/б-17-2-о
Горбенко К. Н.
Проверил
Дрозин А.Ю.

Севастополь
2020

1 ЦЕЛЬ РАБОТЫ

Исследовать алгоритмическое построение методов взаимодействия распределено выполняющихся процессов.

2 ПОСТАНОВКА ЗАДАЧИ

Вариант №1. Осуществить построение топологии кластера требуемого вида 1; выполнить широковещательную рассылку вводимого с клавиатуры сообщения от узла S на все остальные узлы. На узле, инициирующем рассылку, выводить (в виде матрицы) топологию и остовое дерево, на остальных хостах кластера после получения сообщения выводить номер хоста и сам текст сообщения.

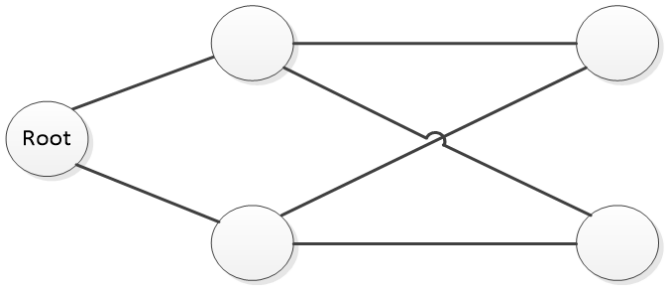

Рисунок 1 – Схема каналов взаимодействия процессов в кластере

3 ХОД РАБОТЫ

Текст программы:

```
#include <mpi.h>
#include <iostream>
#include <fstream>

#define MESSAGE_LEN 1001

using namespace std;

MPI_Status status;
char *message;
int *fullMatrix;
int *skeletonMatrix;

int index(int i, int j, int n) {
    return i * n + j;
}

```

```
18 void init(int n) {
19     ifstream f_in("full.txt");
20     ifstream s_in("skeleton.txt");
21     for (int i = 0; i < n; i++) {
22         for (int j = 0; j < n; j++) {
23             f_in >> fullMatrix[index(i, j, n)];
24             s_in >> skeletonMatrix[index(i, j, n)];
25         }
26     }
27     f_in.close();
28     s_in.close();
29 }
30
31 void print(int *matrix, int n) {
32     for (int i = 0; i < n; i++) {
33         for (int j = 0; j < n; j++) {
34             cout << matrix[index(i, j, n)] << " ";
35         }
36         cout << endl;
37     }
38 }
39
40 void master(int size, int rank) {
41     init(size);
42
43     MPI_Barrier(MPI_COMM_WORLD);
44     MPI_Bcast(fullMatrix, size * size, MPI_INT, 0, MPI_COMM_WORLD);
45
46     MPI_Barrier(MPI_COMM_WORLD);
47     MPI_Bcast(skeletonMatrix, size * size, MPI_INT, 0, MPI_COMM_WORLD);
48
49     cout << "Please input message: (max " << (MESSAGE_LEN - 1) / 2 << ")" <<
        endl;
50     message = new char[MESSAGE_LEN];
51     cin.getline(message, MESSAGE_LEN);
52     cout << endl;
53
54     int sendCount = 0;
55     for (int i = 0; i < size; i++) {
56         if (skeletonMatrix[index(rank, i, size)] == 1) {
57             MPI_Send(message, MESSAGE_LEN, MPI_CHAR, i, 0, MPI_COMM_WORLD);
58             sendCount++;
59         }
60     }
61
62     while (sendCount) {
63         MPI_Recv(NULL, 0, MPI_INT, MPI_ANY_SOURCE, 1, MPI_COMM_WORLD, &status);
64         sendCount--;
```

```
65      }
66
67      cout << endl << "Full matrix: " << endl;
68      print(fullMatrix, size);
69      cout << endl << "Skeleton matrix: " << endl;
70      print(skeletonMatrix, size);
71  }
72
73  void slave(int size, int rank) {
74      MPI_Barrier(MPI_COMM_WORLD);
75      MPI_Bcast(fullMatrix, size * size, MPI_INT, 0, MPI_COMM_WORLD);
76
77      MPI_Barrier(MPI_COMM_WORLD);
78      MPI_Bcast(skeletonMatrix, size * size, MPI_INT, 0, MPI_COMM_WORLD);
79
80      message = new char[MESSAGE_LEN];
81      MPI_Recv(message, MESSAGE_LEN, MPI_CHAR, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD,
            &status);
82      cout << "r[" << status.MPI_SOURCE << "] -> r[" << rank << "]: '" << message
            << "'" << endl;
83
84      int countSends = 0;
85      for (int i = 0; i < size; i++) {
86          if (skeletonMatrix[index(rank, i, size)] == 1) {
87              MPI_Send(message, MESSAGE_LEN, MPI_CHAR, i, 0, MPI_COMM_WORLD);
88              countSends++;
89          }
90      }
91
92      for (; countSends; countSends--) {
93          MPI_Recv(NULL, 0, MPI_INT, MPI_ANY_SOURCE, 1, MPI_COMM_WORLD, NULL);
94      }
95      MPI_Send(NULL, 0, MPI_INT, status.MPI_SOURCE, 1, MPI_COMM_WORLD);
96  }
97
98  int main(int argc, char **argv) {
99      int size, rank;
100     MPI_Init(&argc, &argv);
101     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
102     MPI_Comm_size(MPI_COMM_WORLD, &size);
103
104     fullMatrix = new int[size * size];
105     skeletonMatrix = new int[size * size];
106     !rank ? master(size, rank) : slave(size, rank);
107
108     MPI_Finalize();
109     return 0;
110 }
```

На рисунке 2 представлен результат работы программы:

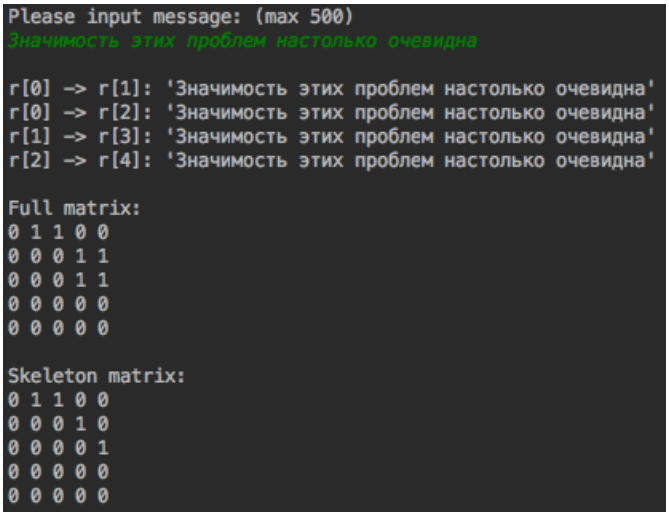

Рисунок 2 – Результат работы программы

ВЫВОДЫ

В ходе выполнения лабораторной работы были исследованы алгоритмические методы построения взаимодействия распределено выполняющихся процессов, закреплены практические навыки построения модели «зонд-эхо», «распределенных семафоров» и «передача маркера».