

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.

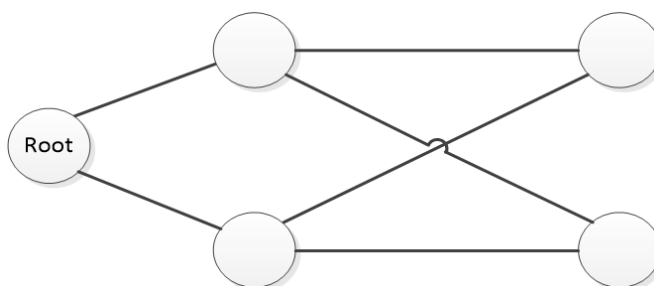


Рисунок 1 – Схема каналов взаимодействия процессов в кластере

## 3 ХОД РАБОТЫ

Текст программы:

```

1 #include <mpi.h>
2 #include <iostream>
3 #include <fstream>
4
5 #define MESSAGE_LEN 1001
6
7 using namespace std;
8
9 MPI_Status status;
10 char *message;
11 int *fullMatrix;
12 int *skeletonMatrix;
13
14 int index(int i, int j, int n) {
15     return i * n + j;
16 }
17

```

```

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,
82             &status);
83     cout << "r[" << status.MPI_SOURCE << "] -> r[" << rank << "]: '" << message
84         << "'" << endl;
85
86     int countSends = 0;
87     for (int i = 0; i < size; i++) {
88         if (skeletonMatrix[index(rank, i, size)] == 1) {
89             MPI_Send(message, MESSAGE_LEN, MPI_CHAR, i, 0, MPI_COMM_WORLD);
90             countSends++;
91         }
92     }
93
94     for (; countSends; countSends--) {
95         MPI_Recv(NULL, 0, MPI_INT, MPI_ANY_SOURCE, 1, MPI_COMM_WORLD, NULL);
96     }
97     MPI_Send(NULL, 0, MPI_INT, status.MPI_SOURCE, 1, MPI_COMM_WORLD);
98 }
99
100 int main(int argc, char **argv) {
101     int size, rank;
102     MPI_Init(&argc, &argv);
103     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
104     MPI_Comm_size(MPI_COMM_WORLD, &size);
105
106     fullMatrix = new int[size * size];
107     skeletonMatrix = new int[size * size];
108     !rank ? master(size, rank) : slave(size, rank);
109
110     MPI_Finalize();
111     return 0;
112 }

```

На рисунке 2 представлен результат работы программы:

```
Please input message: (max 500)
Значимость этих проблем настолько очевидна

r[0] -> r[1]: 'Значимость этих проблем настолько очевидна'
r[0] -> r[2]: 'Значимость этих проблем настолько очевидна'
r[1] -> r[3]: 'Значимость этих проблем настолько очевидна'
r[2] -> r[4]: 'Значимость этих проблем настолько очевидна'

Full matrix:
0 1 1 0 0
0 0 0 1 1
0 0 0 1 1
0 0 0 0 0
0 0 0 0 0

Skeleton matrix:
0 1 1 0 0
0 0 0 1 0
0 0 0 0 1
0 0 0 0 0
0 0 0 0 0
```

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».