МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федерально автономное бюджетное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационных систем

Куркчи Ариф Эрнестович

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-31-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Теория распределенных систем и параллельных вычислений»

на тему «Исследование моделей взаимодействия распределенно выполняющихся процессов»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

лорд ситх   Н. В. Балясный

(должность) (подпись) (инициалы, фамилия)

Севастополь 2016

1. Цель работы

## Исследовать алгоритмическое построение методов взаимодействия распределено выполняющихся процессов.

2. Постановка задачи

Вариант №1

Осуществить построение топологии кластера требуемого вида (рис. 2.1); 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 2.1 – Схема каналов взаимодействия процессов в кластере

3. 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, &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 | } | |  |

4. Результаты

На рисунке 4.1 представлен скриншот выполнения написанной программы.

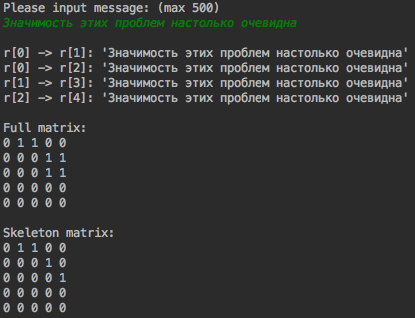


Рисунок 4.1 – Тестовый запуск программы

Вывод

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».