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

Отчет
по лабораторной работе №1
«Исследование средств создания распределено выполняющихся программ»
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 функции библиотеки MPI, необходимые для создания и взаимодействия распределено выполняемых программ.

2 ПОСТАНОВКА ЗАДАЧИ

Вариант №1. Программа осуществляет умножение двух матриц. Размеры матриц – $3 * 3$ и $4 * 4$. На каждом процессе, определяет произведение одной строки первой матрицы на все столбцы второй матрицы. Результаты возвращаются в родительскую задачу.

3 ХОД РАБОТЫ

Исходный код программы представлен на листинге:

```
1 #include <iostream>
2 #include <fstream>
3 #include <mpi.h>
4
5 using namespace std;
6
7 MPI_Status status;
8
9 void master_send(int **m_a, int **m_b, int n, int cor, int i, int j, int comm)
      {
10     int buf[100];
11     buf[0] = n;
12     buf[1] = cor;
13     for (int l = 0; l < n; l++) {
14         buf[2 + l] = m_a[i][l];
15     }
16     for (int l = 0; l < n; l++) {
17         buf[2 + n + l] = m_b[l][j];
18     }
19     MPI_Send(buf, 2 * n + 2, MPI_INT, comm, 1, MPI_COMM_WORLD);
20 }
21
22 void master() {
23     int size;
24     MPI_Comm_size(MPI_COMM_WORLD, &size);
25     if (size == 1) {
26         cout << "Can‘t run without slaves! Just buy some..." << endl;
27         return;
28     }
```

```
29      ifstream is("input.txt");
30
31      int n, m, k;
32      is >> n >> m >> k;
33
34      int **m_a = new int *[n];
35      int **m_b = new int *[m];
36      for (int i = 0; i < n; i++) {
37          m_a[i] = new int[m];
38          for (int j = 0; j < m; j++) {
39              is >> m_a[i][j];
40          }
41      }
42      for (int i = 0; i < m; i++) {
43          m_b[i] = new int[k];
44          for (int j = 0; j < k; j++) {
45              is >> m_b[i][j];
46          }
47      }
48      is.close();
49
50
51      bool used[size];
52      memset(used, 0, sizeof(used));
53      used[0] = true;
54      long long m_c[n][k];
55      int len = n * k;
56      int j = 0;
57      int online = 0;
58      for (int i = 1; i < size && j < len; i++) {
59          master_send(m_a, m_b, m, j, j / k, j % k, i);
60          used[i] = true;
61          online++;
62          j++;
63      }
64
65      long message[2];
66      while (j < len || online > 0) {
67          MPI_Recv(message, 2, MPI_LONG_LONG, MPI_ANY_SOURCE, 1, MPI_COMM_WORLD,
                &status);
68          m_c[message[0] / k][message[0] % k] = message[1];
69          used[status.MPI_SOURCE] = false;
70          online--;
71
72          if (j < len) {
73              master_send(m_a, m_b, m, j, j / k, j % k, status.MPI_SOURCE);
74              used[status.MPI_SOURCE] = true;
75              online++;
```

```
76              j++;
77          }
78      }
79
80      for (int i = 1; i < size; i++) {
81          int buf[1];
82          MPI_Send(buf, 0, MPI_LONG_LONG, i, 2, MPI_COMM_WORLD);
83      }
84
85      for (int i = 0; i < n; i++) {
86          for (int j = 0; j < k; j++) {
87              cout << m_c[i][j] << " ";
88          }
89          cout << endl;
90      }
91  }
92
93  void slave() {
94      int message[100];
95      bool running = true;
96      while (running) {
97          MPI_Recv(message, 100, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &status
                );
98          if (status.MPI_TAG == 2) {
99              running = false;
100         } else {
101             int n = message[0];
102             long long result[2] = {message[1], 0};
103             for (int i = 0; i < n; i++) {
104                 result[1] += message[2 + i] * message[2 + n + i];
105             }
106             MPI_Send(result, 2, MPI_LONG_LONG, 0, 1, MPI_COMM_WORLD);
107         }
108     }
109 }
110
111 int main(int argc, char **argv) {
112     int rank;
113
114     MPI_Init(&argc, &argv);
115     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
116
117     rank ? slave() : master();
118
119     MPI_Barrier(MPI_COMM_WORLD);
120     MPI_Finalize();
121     return 0;
122 }
```

ВЫВОД

В ходе данной лабораторной работы были изучены основные принципы библиотеки MPI и ее функции. Написана программа, осуществляющая распределённые вычисления на любом количества хостов (не менее 2х), в которой один из хостов – главный, а все остальные – подчиненные. Главный процесс распределяет задачи по умножению между подчиненными, которые осуществляют умножение. После выполнения задачи всем подчиненным процессам отправляется пустое сообщение с определённым тегом для завершения их ожидания.

Отмечу, что использование количества процессов, равное произведению количества строк первой матрицы на количество столбцов второй + 1 является наиболее эффективным, так как все задачи распределяются одновременно и главный процесс не вынужден ожидать момента, когда один из процессов освободится для распределения следующей части задания. При этом большее количество процессов будет избыточным, так как каждый последующий процесс в итоге так и не будет задействован.