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 №1

по дисциплине «Теория распределенных систем и параллельных вычислений»

на тему «Исследование средство создания распределено выполняющихся программ»

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

(дата)

Руководитель практикума

ст. преподаватель   А. Ю. Дрозин

(должность) (подпись) (инициалы, фамилия)

Севастополь 2016

1. ЦЕЛЬ РАБОТЫ

Исследовать функции библиотеки MPI, необходимые для создания и взаимодействия распределено выполняемых программ.

1. 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.

1. ХОД РАБОТЫ

Исходный код программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 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 | } | |  |

ВЫВОДЫ

В ходи данной лабораторной работы были изучены основные принципы и функции для работы с библиотекой MPI. Написана программа, осуществляющая распределённые вычисления на любом количества хостов (не менее 2х), в которой один из хостов будет представлять собой мастер-процесс, а все остальные слейв-процессы. Мастер процесс распределяет задачи по умножению между слейвами, которые занимаются только умножением. По выполнению задачи всем слейвам отправляется пустое сообщение с определённым тегом для завершения их ожидания. Было отмечено, что использование количества процессов, равное произведению количества строк первой матрицы на количество столбцов второй + 1 мастер процесс является наиболее эффективным, так как все задачи распределяются одновременно и мастер не вынужден ожидать момента, когда один из них 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.