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 №2

по дисциплине «Теория распределенных систем и параллельных вычислений»

на тему «Исследование коллективного типа передачи данных, групп и коммуникаторов в MPI»

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

(дата)

Руководитель практикума

ассистент   Н. В. Балясный

(должность) (подпись) (инициалы, фамилия)

Севастополь 2016

1. ЦЕЛЬ РАБОТЫ

Исследовать способы обмена данными между процессами в режиме широковещания или группового обмена с использованием MPI-функций.

1. ПОСТАНОВКА ЗАДАЧИ

Вариант №1

Реализовать блочный алгоритм распределенного параллельного перемножения матриц и с размерами (8\*5) и (5\*3) соответственно. Вид распределяемых между процессами блоков представлен на рисунке 2.13:



Рисунок 2.1 – Перемножение матриц

Корневой процесс реализует рассылку:

1. блоков матрицы *A* по две строки;
2. широковещательную рассылку элементов матрицы *B* между обрабатывающими процессами внутри своей группы (по умолчанию) – режим One-To-All.

Организуется четыре процесса, обрабатывающих данные и формирующих фрагменты (2\*3) матрицы результата C. После подготовки блоков матрицы всеми обрабатывающими процессами (взаимная синхронизация функцией MPI\_Barrier) выполняется совместная передача результатов (блоков матрицы C) корневому процессу – режим All-To-One.

1. ХОД РАБОТЫ

Исходный код программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 | #include <iostream> | | 2 | #include <fstream> | | 3 | #include <mpi.h> | | 4 | #include <iomanip> | | 5 |  | | 6 | using namespace std; | | 7 |  | | 8 | *int* len\_for\_node(*int* *size*, *int* *all*) { | | 9 | if (size == 1) { | | 10 | return all; | | 11 | } | | 12 | if (!(all % size)) { | | 13 | return all / size; | | 14 | } | | 15 | return 0; | | 16 | } | | 17 |  | | 18 | *void* master() { | | 19 | *int* size; | | 20 | MPI\_Comm\_size(MPI\_COMM\_WORLD, &size); | | 21 | if (size == 1) { | | 22 | cout << "Can`t run without slaves! Just buy some..." << endl; | | 23 | return; | | 24 | } | | 25 | ifstream is("input.txt"); | | 26 | *int* n, k, m; | | 27 | is >> n >> k >> m; | | 28 | *int* \*init\_data = new *int*[4]; | | 29 | init\_data[0] = n; | | 30 | init\_data[1] = k; | | 31 | init\_data[2] = m; | | 32 |  | | 33 | *int* len = len\_for\_node(size - 1, n); | | 34 | init\_data[3] = len; | | 35 | if (!len) { | | 36 | cout << "Can`t split lines by processes" << endl; | | 37 | exit(418); | | 38 | } | | 39 |  | | 40 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 41 | MPI\_Bcast(init\_data, 4, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 42 | delete[] init\_data; | | 43 |  | | 44 | *int* \*a = new *int*[n \* k], | | 45 | \*b = new *int*[k \* m]; | | 46 | for (*int* i = 0; i < n \* k; i++) { | | 47 | is >> a[i]; | | 48 | } | | 49 | for (*int* i = 0; i < k \* m; i++) { | | 50 | is >> b[i]; | | 51 | } | | 52 |  | | 53 | *int* \*empty = new *int*[len \* k]; | | 54 | *int* \*buf = new *int*[(len + n) \* k]; | | 55 | memcpy(buf + (len \* k), a, n \* k \* sizeof(*int*)); | | 56 |  | | 57 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 58 | MPI\_Scatter(buf, len \* k, MPI\_INTEGER, empty, len \* k, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 59 | delete[] empty; | | 60 | delete[] buf; | | 61 | delete[] a; | | 62 |  | | 63 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 64 | MPI\_Bcast(b, k \* m, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 65 | delete[] b; | | 66 |  | | 67 | *int* \*c = new *int*[(len + n) \* m]; | | 68 | empty = new *int*[len \* m]; | | 69 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 70 | MPI\_Gather(empty, len \* m, MPI\_INTEGER, c, len \* m, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 71 |  | | 72 | for (*int* i = 0; i < n; i++) { | | 73 | for (*int* j = 0; j < m; j++) { | | 74 | cout << setw(5) << c[len \* m + i \* m + j] << " "; | | 75 | } | | 76 | cout << endl; | | 77 | } | | 78 | delete[] c; | | 79 | } | | 80 |  | | 81 | *void* slave() { | | 82 | *int* \*buf = new *int*[4]; | | 83 |  | | 84 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 85 | MPI\_Bcast(buf, 4, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 86 | *int* k = buf[1], | | 87 | m = buf[2], | | 88 | len = buf[3]; | | 89 | delete[] buf; | | 90 |  | | 91 | *int* empty[0]; | | 92 | *int* \*a = new *int*[len \* k]; | | 93 |  | | 94 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 95 | MPI\_Scatter(empty, 0, MPI\_INTEGER, a, len \* k, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 96 |  | | 97 | *int* \*b = new *int*[k \* m]; | | 98 |  | | 99 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 100 | MPI\_Bcast(b, k \* m, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 101 |  | | 102 | *int* \*c = new *int*[len \* m]; | | 103 | for (*int* i = 0; i < len; i++) { | | 104 | for (*int* j = 0; j < m; j++) { | | 105 | c[i \* m + j] = 0; | | 106 | for (*int* y = 0; y < k; y++) { | | 107 | c[i \* m + j] += a[i \* k + y] \* b[y \* m + j]; | | 108 | } | | 109 | } | | 110 | } | | 111 | delete[] a; | | 112 | delete[] b; | | 113 |  | | 114 | MPI\_Barrier(MPI\_COMM\_WORLD); | | 115 | MPI\_Gather(c, len \* m, MPI\_INTEGER, empty, 0, MPI\_INTEGER, 0, MPI\_COMM\_WORLD); | | 116 | delete[] c; | | 117 | } | | 118 |  | | 119 | *int* main(*int* *argc*, *char* \*\**argv*) { | | 120 | *int* rank; | | 121 |  | | 122 | MPI\_Init(&argc, &argv); | | 123 | MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); | | 124 |  | | 125 | rank ? slave() : master(); | | 126 |  | | 127 | MPI\_Finalize(); | | 128 | return 0; | | 129 | } | |  | |

ВЫВОДЫ

В ходе выполнения лабораторной работы были исследованы способы обмена данными между процессами в режиме широковещания или группового обмена с использованием MPI-функций. Написана программа умножения двух матриц произвольного размера, распределяющая расчет строк динамически между кратным количеством процессов-рабочих.