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 №4

по дисциплине «Теория распределенных систем и параллельных вычислений»

на тему «Исследование взаимодействий распределенных процессов типа Клиент-сервер»

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

(дата)

Руководитель практикума

лорд ситх   Н. В. Балясный

(должность) (подпись) (инициалы, фамилия)

Севастополь 2016

1. Цель работы

Исследовать механизм взаимодействия распределено выполняющихся параллельных процессов типа «клиент-сервер».

2. Постановка задачи

В состав вычислительного кластера входит три хоста, один из которых реализует функции сервера, два остальных – клиентов.

Сервер разграничивает доступ к трем общим ресурсам – нерешенным, хранящим общую вырученную сумму от продажи товаров, общее количество товаров и остальных товаров. Доступ к ресурсам осуществляется в произвольном порядке, все ресурсы разделяются между клиентами по отдельности. Реализована процедура, выделяющая ресурсы (путем передачи сообщения) в использование клиентам. Реализовать серверный процесс, который разграничивает доступ клиентов к этой процедуре (процедурам) и к ресурсам. Реализацию сервера выполнять в соответствии со схемой управления, использующую рассылку сообщений.

3. Текст программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | #include <mpi.h> | | 2 | #include <iostream> | | 3 | #include <sstream> | | 4 | #include <iomanip> | | 5 |  | | 6 | #define CS\_CONNECT 0 | | 7 | #define CS\_DISCONNECT 1 | | 8 | #define CS\_TAKE 2 | | 9 | #define CS\_RETURN 3 | | 10 |  | | 11 | #define SC\_FREE 10 | | 12 | #define SC\_INUSE 11 | | 13 | #define SC\_SET 12 | | 14 | #define SC\_NO\_RESOURCE 13 | | 15 | #define SC\_NO\_ACTION 14 | | 16 |  | | 17 | #define RES\_COUNT 3 | | 18 |  | | 19 | using namespace std; | | 20 |  | | 21 | *void* server() { | | 22 | *double* resources[] = {1234.56, 12, 500}; | | 23 | *int* inUse[] = {0, 0, 0}; | | 24 |  | | 25 | *bool* firstClientConnected = false; | | 26 | *int* clients = 0, outTag, r\_index; | | 27 | *double* r\_value; | | 28 |  | | 29 | *double* \*inBuf = new *double*[2]; | | 30 | *double* \*outBuf = new *double*[1]; | | 31 | MPI\_Status status; | | 32 | while (!firstClientConnected || clients) { | | 33 | outBuf[0] = 0; | | 34 | MPI\_Recv(inBuf, 2, MPI\_DOUBLE, MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status); | | 35 | switch (status.MPI\_TAG) { | | 36 | case CS\_CONNECT: | | 37 | firstClientConnected = true; | | 38 | clients++; | | 39 | break; | | 40 | case CS\_DISCONNECT: | | 41 | clients--; | | 42 | break; | | 43 | case CS\_TAKE: | | 44 | r\_index = (*int*) inBuf[0]; | | 45 | if (r\_index < RES\_COUNT) { | | 46 | if (inUse[r\_index] != 0) { | | 47 | outTag = SC\_INUSE; | | 48 | } else { | | 49 | outBuf[0] = resources[r\_index]; | | 50 | inUse[r\_index] = status.MPI\_SOURCE; | | 51 | outTag = SC\_FREE; | | 52 | } | | 53 | } else { | | 54 | outTag = SC\_NO\_RESOURCE; | | 55 | } | | 56 | MPI\_Send(outBuf, 1, MPI\_DOUBLE, status.MPI\_SOURCE, outTag, MPI\_COMM\_WORLD); | | 57 | break; | | 58 | case CS\_RETURN: | | 59 | r\_index = (*int*) inBuf[0]; | | 60 | r\_value = inBuf[1]; | | 61 | if (r\_index < RES\_COUNT) { | | 62 | if (inUse[r\_index] != status.MPI\_SOURCE) { | | 63 | outTag = SC\_INUSE; | | 64 | } else { | | 65 | resources[r\_index] = r\_value; | | 66 | inUse[r\_index] = 0; | | 67 | outTag = SC\_SET; | | 68 | } | | 69 | } else { | | 70 | outTag = SC\_NO\_RESOURCE; | | 71 | } | | 72 | MPI\_Send(outBuf, 1, MPI\_DOUBLE, status.MPI\_SOURCE, outTag, MPI\_COMM\_WORLD); | | 73 | break; | | 74 | default: | | 75 | MPI\_Send(outBuf, 1, MPI\_DOUBLE, status.MPI\_SOURCE, SC\_NO\_ACTION, MPI\_COMM\_WORLD); | | 76 | } | | 77 | } | | 78 |  | | 79 | stringstream ss; | | 80 | ss << "\t" << "Server" << endl; | | 81 | for (r\_index = 0; r\_index < RES\_COUNT; r\_index++) { | | 82 | ss << "Resource #" << r\_index | | 83 | << " is " << fixed << setw(8) << setprecision(2) << resources[r\_index] | | 84 | << endl; | | 85 | } | | 86 | ss << endl; | | 87 | cout << ss.str(); | | 88 |  | | 89 | delete[] inBuf; | | 90 | delete[] outBuf; | | 91 | } | | 92 |  | | 93 | *double* get\_resource(*int* *n*) { | | 94 | MPI\_Status status; | | 95 | *double* \*inBuf = new *double*[1]; | | 96 | *double* \*outBuf = new *double*[1]; | | 97 | outBuf[0] = n; | | 98 | *bool* received = false; | | 99 | while (!received) { | | 100 | MPI\_Send(outBuf, 1, MPI\_DOUBLE, 0, CS\_TAKE, MPI\_COMM\_WORLD); | | 101 | MPI\_Recv(inBuf, 1, MPI\_DOUBLE, 0, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status); | | 102 | if (status.MPI\_TAG == SC\_FREE) { | | 103 | received = true; | | 104 | } | | 105 | } | | 106 | *double* ret\_val = inBuf[0]; | | 107 | delete[] inBuf; | | 108 | delete[] outBuf; | | 109 |  | | 110 | return ret\_val; | | 111 | } | | 112 |  | | 113 | *bool* set\_resource(*int* *n*, *double* *value*) { | | 114 | MPI\_Status status; | | 115 | *double* \*inBuf = new *double*[1]; | | 116 | *double* \*outBuf = new *double*[2]; | | 117 | outBuf[0] = n; | | 118 | outBuf[1] = value; | | 119 | MPI\_Send(outBuf, 2, MPI\_DOUBLE, 0, CS\_RETURN, MPI\_COMM\_WORLD); | | 120 | MPI\_Recv(inBuf, 1, MPI\_DOUBLE, 0, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status); | | 121 |  | | 122 | delete[] inBuf; | | 123 | delete[] outBuf; | | 124 |  | | 125 | return status.MPI\_TAG == SC\_SET; | | 126 | } | | 127 |  | | 128 | *void* client(*int* *rank*) { | | 129 | *double* \*outBuf = new *double*[1]; | | 130 | outBuf[0] = 0; | | 131 | MPI\_Send(outBuf, 1, MPI\_DOUBLE, 0, CS\_CONNECT, MPI\_COMM\_WORLD); | | 132 |  | | 133 | stringstream ss; | | 134 | ss << "\t" << "Client #" << rank << endl; | | 135 | *double* r\_value, r\_value\_changed; | | 136 | for (*int* r\_index = 0; r\_index < RES\_COUNT; r\_index++) { | | 137 | r\_value = get\_resource(r\_index); | | 138 | r\_value\_changed = r\_value + (r\_index + 10) \* rank; | | 139 | set\_resource(r\_index, r\_value\_changed); | | 140 |  | | 141 | ss << "Resource #" << r\_index | | 142 | << " was " << fixed << setw(8) << setprecision(2) << r\_value | | 143 | << " set " << fixed << setw(8) << setprecision(2) << r\_value\_changed | | 144 | << endl; | | 145 | } | | 146 | ss << endl; | | 147 | cout << ss.str(); | | 148 |  | | 149 | MPI\_Send(outBuf, 1, MPI\_DOUBLE, 0, CS\_DISCONNECT, MPI\_COMM\_WORLD); | | 150 |  | | 151 | delete[] outBuf; | | 152 | } | | 153 |  | | 154 | *int* main(*int* *argc*, *char* \*\**argv*) { | | 155 | *int* rank, size; | | 156 | MPI\_Init(&argc, &argv); | | 157 | MPI\_Comm\_size(MPI\_COMM\_WORLD, &size); | | 158 | MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); | | 159 | if (size != 3) { | | 160 | if (rank == 0) { | | 161 | cout << "Use only with 3 processes" << endl; | | 162 | cout << "Exit..." << endl; | | 163 | } | | 164 | MPI\_Finalize(); | | 165 | return 1; | | 166 | } | | 167 | !rank ? server() : client(rank); | | 168 |  | | 169 | MPI\_Finalize(); | | 170 | return 0; | | 171 | } | |  |

4. Результат

На рисунке 4.1 представлен скриншот демонстрирующий работу написанной программы.

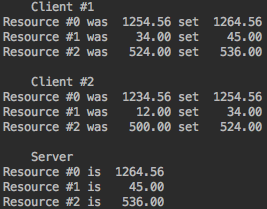


Рисунок 4.1 – Тестовый пример

Вывод

В ходе лабораторной работы был исследован механизм взаимодействия распределено выполняющихся параллельных процессов типа «клиент-сервер».