Ковальский Евгений

Лабораторная работа №1

Задание - Вариант 0.

Реализуйте функцию ring, которая создаёт N процессов и посылает сообщение первому процессу, который посылает сообщение второму, второй - третьему, и так далее. Наконец, процесс N посылает сообщение обратно процессу 1. После того, как сообщение обежало вокруг кольца M раз, все процессы заканчивают работу.

**Задание.** Модифицировать программу, написанную на Л.Р. №1 так чтобы она работала на основе обобщенной передачи сообщений. Результаты работы сравнить (с результатами полученными в Л.Р. №2) и занести в отчет.

Код программы:

*//--------------------Подключаемые* *библиотеки--------------------//*

#include <headers/mpi.h>

#include <iostream>

#include <cstring>

#include <chrono>

#include <time.h>

*using* *namespace* std;

int **main**(int argc, char\* argv[])

{

*//--------------------Иициализация--------------------//*

int sizeX = 100;

int procNum, procRank;

int m = 5;*//кол-во* *итераций*

MPI\_Init(&argc, *&argv*);

MPI\_Comm\_size(MPI\_COMM\_WORLD, *&procNum*);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, *&procRank*);

*//семя* *для* *функции* *rand(),* *смещено,* *чтобы* *было* *разное* *у* *разных* *процессов*

srand(time(0)+procRank + 314159 \* 451 % 42);

*auto* start = std::chrono::steady\_clock::now();

*//--------------------Инициализация* *буфферов--------------------//*

*const* int numMsg = 2;

int sendBuff[procNum][numMsg];

int\* recv = *new* int[numMsg];

*//--------------------Главный* *цикл--------------------//*

*for*(int j = 0; j < m; ++j){

*for*(int i = 0; i < procNum; ++i) {

int mes1 = i+1;

int mes2 = rand() % 100;

*for*(int k = 0; k < procNum; ++k){

sendBuff[k][0] = mes1;

sendBuff[k][1] = mes2;

}

*if*(procRank == i){

cout << "processor " << procRank << " send message " << mes2 << endl;

}

MPI\_Scatter(sendBuff, numMsg, MPI\_INT, *recv*, numMsg, MPI\_INT, i, MPI\_COMM\_WORLD);

*if*(recv[0] == procRank) {

cout << "processor " << procRank << " recv message " << recv[1] << endl;

}

*if*(recv[0] == procNum && procRank == 0) {

cout << "processor " << procRank << " recv message " << recv[1] << endl;

}

}

}

*delete*[] recv;

*//--------------------Завершение* *работы* *MPI--------------------//*

MPI\_Finalize();

*auto* end = std::chrono::steady\_clock::now();

std::chrono::duration<double> elapsed\_seconds = end-start;

std::cout << "elapsed time: " << elapsed\_seconds.count() << "s\n";

*return* 0;

}

Вывод программы (4 процесса):

processor 0 send message 40

processor 1 recv message 40

processor 1 send message 34

processor 2 recv message 34

processor 2 send message 75

processor 3 recv message 75

processor 3 send message 92

processor 0 recv message 92

processor 0 send message 94

processor 1 recv message 94

processor 1 send message 61

processor 2 recv message 61

processor 2 send message 55

processor 3 recv message 55

processor 3 send message 22

processor 0 recv message 22

processor 0 send message 84

processor 1 recv message 84

processor 1 send message 76

processor 2 recv message 76

processor 2 send message 16

processor 3 recv message 16

processor 3 send message 18

processor 0 recv message 18

processor 0 send message 37

processor 1 recv message 37

processor 1 send message 31

processor 2 recv message 31

processor 2 send message 73

processor 3 recv message 73

processor 3 send message 38

processor 0 recv message 38

processor 0 send message 83

processor 1 recv message 83

processor 1 send message 19

processor 2 recv message 19

processor 2 send message 31

processor 3 recv message 31

processor 3 send message 68

processor 0 recv message 68

Код программы из Л.Р. №2:

*//--------------------Подключаемые* *библиотеки--------------------//*

#include <headers/mpi.h>

#include <iostream>

#include <cstring>

#include <chrono>

#include <time.h>

*using* *namespace* std;

int **main**(int argc, char\* argv[])

{

*//--------------------Иициализация--------------------//*

int sizeX = 100;

int procNum, procRank, recv;

int m=5;*//кол-во* *итераций*

MPI\_Init(&argc, *&argv*);

MPI\_Comm\_size(MPI\_COMM\_WORLD, *&procNum*);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, *&procRank*);

*//семя* *для* *функции* *rand(),* *смещено,* *чтобы* *было* *разное* *у* *разных* *процессов*

srand(time(0)+procRank + 314159 \* 451 % 42);

*auto* start = std::chrono::steady\_clock::now();

*//--------------------Главный* *цикл--------------------//*

*for*(int j = 0; j < m; ++j){

*for*(int i = 0; i < procNum; ++i) {

int globalSum = 0;

int localSum = 0;

localSum = rand()%100;

MPI\_Reduce(&localSum, *&globalSum*, 1, MPI\_INT, MPI\_SUM, i, MPI\_COMM\_WORLD);

*if*(globalSum != 0){

cout << "Process " << procRank << " recieved " << globalSum << endl;

}

}

}

*//--------------------Завершение* *работы* *MPI--------------------//*

MPI\_Finalize();

*auto* end = std::chrono::steady\_clock::now();

std::chrono::duration<double> elapsed\_seconds = end-start;

std::cout << "elapsed time: " << elapsed\_seconds.count() << "s\n";

*return* 0;

}

Вывод программы (4 процесса):

Process 0 recieved 141

Process 1 recieved 100

Process 2 recieved 220

Process 3 recieved 133

Process 0 recieved 164

Process 1 recieved 253

Process 2 recieved 194

Process 3 recieved 166

Process 0 recieved 174

Process 1 recieved 156

Process 2 recieved 260

Process 3 recieved 248

Process 0 recieved 138

Process 1 recieved 238

Process 2 recieved 186

Process 3 recieved 113

Process 0 recieved 260

Process 1 recieved 151

Process 2 recieved 191

Process 3 recieved 148

Замеры времени работы программы из 2 лабораторной (4 процесса, вывод отключён):

elapsed time: 0.0033851s

elapsed time: 0.0034404s

elapsed time: 0.00341s

elapsed time: 0.0034029s

Замеры времени работы программы из 3 лабораторной (4 процесса, вывод отключён):

elapsed time: 0.0026308s

elapsed time: 0.0026887s

elapsed time: 0.0026907s

elapsed time: 0.0027487s