МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №6 по дисциплине «Параллельное программирование»

**Основы MPI. Парные обмены**

|  |  |
| --- | --- |
| Выполнил: студент группы ФИб-4302-51-00 | / Д.А. Савин / |
| Проверил: ст. преподаватель каф. ПМиИ | / В.А. Бызов / |

Киров 2021

Задание 1

Напишите программу, в которой каждый процесс выводит на экран свой номер и общее количество процессов в формате:

I am < Номер процесса > process from < Количество процессов > processes!

Код:

|  |
| --- |
| #include <iostream>  #include <math.h>  #include <mpi.h>  int main(int argc, char\* argv[])  {  int proc\_rank, proc\_num;  const int buf\_size = 100;  char buf[buf\_size];  MPI\_Status st;  MPI\_Init(&argc, &argv);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &proc\_rank);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);  printf\_s("I am %d process from %d processes", proc\_rank, proc\_num);  MPI\_Finalize();  } |

Изображение выглядит как текст

Автоматически созданное описание

Задание 2

Напишите программу, в которой каждый процесс с чётным номером выводит на экран строку

<Номер процесса>: FIRST!

а каждый процесс с нечётным номером строку

<Номер процесса>: SECOND!

Нулевой процесс должен вывести на экран информацию о количестве работающих процессов в формате

<Количество процессов> processes.

Код:

|  |
| --- |
| #include <iostream>  #include <math.h>  #include <mpi.h>  int main(int argc, char\* argv[])  {  int proc\_rank, proc\_num;  const int buf\_size = 100;  char buf[buf\_size];  MPI\_Status st;  MPI\_Init(&argc, &argv);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &proc\_rank);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);  if (proc\_rank == 0)  {  for (int i = 1; i < proc\_num; i++)  {  MPI\_Recv(buf, buf\_size, MPI\_CHAR, MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &st);  printf\_s("%s\n", buf);  }  printf\_s("\n%d processes.\n", proc\_num);  }  else  {  if (proc\_rank % 2 == 0)  {  sprintf\_s(buf, "%d FIRST", proc\_rank);  }  else  {  sprintf\_s(buf, "%d SECOND", proc\_rank);  }  MPI\_Send(buf, buf\_size, MPI\_CHAR, 0, 0, MPI\_COMM\_WORLD);  }  MPI\_Finalize();  } |

Изображение выглядит как текст

Автоматически созданное описание

Задание 3

Скомпилируйте и запустите на выполнение приведённый ниже код. Поясните, почему возникла тупиковая ситуация? Исправьте программу, заменив блокирующие вызовы на неблокирующие.

Код:

|  |
| --- |
| #include <iostream>  #include <mpi.h>  #include <vector>  constexpr auto MSGLEN = 32768; //размер сообщения  constexpr auto TAG\_A = 100;  constexpr auto TAG\_B = 200;  using namespace std;  int main(int argc, char\* argv[])  {  vector<float> message1(MSGLEN), message2(MSGLEN); //пересылаемые сообщения  int rank, dest, source, send\_tag, recv\_tag;  MPI\_Status status;  MPI\_Request request;  MPI\_Init(&argc, &argv);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);  for (int i = 0; i < MSGLEN; i++) {  message1[i] = 1 - 2 \* rank;  }  if (rank == 0) {  dest = 1;  source = 1;  send\_tag = TAG\_A;  recv\_tag = TAG\_B;  }  else if (rank == 1) {  dest = 0;  source = 0;  send\_tag = TAG\_B;  recv\_tag = TAG\_A;  }  cout << "Task " << rank << " has sent the message" << endl;  MPI\_Isend(message1.data(), MSGLEN, MPI\_FLOAT, dest, send\_tag,  MPI\_COMM\_WORLD, &request);  MPI\_Irecv(message2.data(), MSGLEN, MPI\_FLOAT, source, recv\_tag,  MPI\_COMM\_WORLD, &request);  cout << " Task " << rank << " has received the message" << endl;  MPI\_Finalize();  return 0;  } |

Изображение выглядит как текст

Автоматически созданное описание

Задание 4

**Эстафетная палочка**. Организовать передачу данных по кругу. Нулевой процесс генерирует случайное целое число и передаёт его первому процессу. Далее каждый процесс получает некоторое целое число, прибавляет к нему 1 и передаёт следующему процессу. Последний процесс передаёт число нулевому. Нулевой процесс выводит «Correct!», если новое число на p – 1 больше исходного или «Error!» в противном случае. Остальные процессы выводят сообщение «<Номер процесса> receive number <число>».

Код:

|  |
| --- |
| #include <iostream>  #include <math.h>  #include <mpi.h>  #include <random>  using namespace std;  int main(int argc, char\* argv[])  {  int proc\_rank, proc\_num;  const int buf\_size = 100;  char buf[buf\_size];  int rand\_num = 0;  int firs\_value = 0;  int last\_value = 333;  MPI\_Status st;  int oldValue;  MPI\_Init(&argc, &argv);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &proc\_rank);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_num);  //MPI\_Recv(buf, buf\_size, MPI\_CHAR, MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &st);  //MPI\_Send(buf, buf\_size, MPI\_CHAR, 0, 0, MPI\_COMM\_WORLD);  if (proc\_rank == 0)  {  rand\_num = firs\_value + rand() % last\_value;  cout << "generated - " << rand\_num << endl << endl;  oldValue = rand\_num;  //cout << proc\_rank << " send to 1 num= " << rand\_num << endl;  sprintf\_s(buf, "%d", rand\_num);  MPI\_Send(buf, buf\_size, MPI\_CHAR, 1, 0, MPI\_COMM\_WORLD);  MPI\_Recv(buf, buf\_size, MPI\_CHAR, (proc\_num - 1), 0, MPI\_COMM\_WORLD, &st);  //cout << proc\_rank << " receive " << buf << endl;  if (atoi(buf) - oldValue == proc\_num - 1)  {  cout << "Correct!" << endl;  }  else  {  cout << "Error!" << endl;  }  }  else  {  MPI\_Recv(buf, buf\_size, MPI\_CHAR, (proc\_rank - 1), 0, MPI\_COMM\_WORLD, &st);  cout << proc\_rank << " receive " << buf << endl;  rand\_num = atoi(buf) + 1;  sprintf\_s(buf, "%d", rand\_num);  if (proc\_rank == proc\_num - 1)  {  //cout << proc\_rank << " send to 0 num= " << rand\_num << endl;  MPI\_Send(buf, buf\_size, MPI\_CHAR, 0, 0, MPI\_COMM\_WORLD);  }  else  {  //cout << proc\_rank << " send to " << proc\_rank + 1 << " num= " << rand\_num << endl;  MPI\_Send(buf, buf\_size, MPI\_CHAR, (proc\_rank + 1), 0, MPI\_COMM\_WORLD);  }  //printf\_s(buf);  }  MPI\_Finalize();  delete[] buf;  } |

Изображение выглядит как текст

Автоматически созданное описание

Задание 5

**Концепция master-slave**. Нулевой процесс генерирует N массивов целых чисел из M элементов. Далее он распределяет по остальным процессам по одному массиву. Каждый процесс, получив массив, считает сумму его элементов и отсылает обратно нулевому. Нулевой процесс добавляет полученный результат к глобальной сумме и отправляет освободившемуся процессу новую работу. Так происходит до тех пор, пока не вычислена сумма всех элементов всех массивов (сумма элементов матрицы). Реализовать алгоритм двумя способами: с использованием блокирующих и неблокирующих операций. Протестировать для следующих пар N и M: 100 и 1000000, 10000 и 10000, 1000000 и 100. Сравнить быстродействие. Сделать выводы.

Код:

|  |
| --- |
| #include <iostream>  #include <mpi.h>  #include <vector>  #include <clocale>  #include <random>  using namespace std;  int main(int argc, char\* argv[])  {  const int N = 4, M = 3, a = 10, b = 100;  MPI\_Init(&argc, &argv);  MPI\_Request request;  MPI\_Status status;  int rank, size;  MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);  int rows = N;  int\*\* array = new int\* [rows];  int size1 = rows \* M;  array[0] = new int[size1]; //инициализируем массив  for (int j = 1; j < rows; j++)  array[j] = &array[0][j \* M];  //локальная сумма для каждого потока  int\* sum = new int;  \*sum = 0;  //общая сумма (сумма всех локальных сумм)  int\* Sum = new int;  \*Sum = 0;  if (rank == 0)  {  random\_device rd;  mt19937 gen(rd());  uniform\_int\_distribution<> dis(a, b);  for (int i = 0; i < N; i++) // 0 поток генерирует массив  {  for (int j = 0; j < M; j++)  {  array[i][j] = dis(gen);  cout << array[i][j] << " ";  }  cout << endl;  }  cout << endl;  for (int i = 0; i < N; i++)  {  if (i % size != 0) //отправка каждому из потоков одномерного массива  {  MPI\_Send(array[i], M, MPI\_INT, i % size, 0, MPI\_COMM\_WORLD);  cout << "process: " << rank << " send row: " << i << endl;  }  else  {  MPI\_Send(array[i], M, MPI\_INT, i % size + 1, 0, MPI\_COMM\_WORLD);  cout << "process: " << rank << " send row: " << i << endl;  }  }  for (int i = 0; i < N; i++) // 0 поток принимает от каждого потока его локальную сумму и прибавляет к общей  {  if (i % size != 0)  {  MPI\_Recv(sum, 1, MPI\_INT, i % size, 0, MPI\_COMM\_WORLD, &status);  cout << "process: " << rank << " receive sum: " << \*sum << endl;  \*Sum += \*sum;  }  else  {  MPI\_Recv(sum, 1, MPI\_INT, i % size + 1, 0, MPI\_COMM\_WORLD, &status);  cout << "process: " << rank << " receive sum: " << \*sum << endl;  \*Sum += \*sum;  }  }  cout << "Sum = " << \*Sum << endl;  }  else  {  for (int i = 0; i < N; i++) //каждый из потоков принимает от 0 часть массива подсчитывает сумму элементов и отправляет обратно  {  MPI\_Recv(array[i], M, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, &status);  for (int j = 0; j < M; j++)  {  cout << "process: " << rank << " recieve number: " << array[i][j] << endl;  \*sum += array[i][j];  }  MPI\_Send(sum, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);  cout << "process: " << rank << " send sum: " << \*sum << endl;  \*sum = 0;  }  }  cout << endl;  MPI\_Finalize();  return 0;  } |

Изображение выглядит как текст

Автоматически созданное описание

Задание 6

**Концепция master-slave**. Нулевой процесс генерирует N массивов целых чисел из M элементов. Далее он распределяет по остальным процессам по одному массиву. Каждый процесс, получив массив, считает сумму его элементов и отсылает обратно нулевому. Нулевой процесс добавляет полученный результат к глобальной сумме и отправляет освободившемуся процессу новую работу. Так происходит до тех пор, пока не вычислена сумма всех элементов всех массивов (сумма элементов матрицы). Реализовать алгоритм двумя способами: с использованием блокирующих и неблокирующих операций. Протестировать для следующих пар N и M: 100 и 1000000, 10000 и 10000, 1000000 и 100. Сравнить быстродействие. Сделать выводы.

Код:

|  |
| --- |
| #include <iostream>  #include <mpi.h>  #include <vector>  #include <clocale>  #include <random>  using namespace std;  int main(int argc, char\* argv[])  {  const int N = 3, M = 3, a = 10, b = 100;  int rows = N;  int\*\* array = new int\* [rows];  for (int j = 0; j < rows; j++)  array[j] = new int[M];  random\_device rd;  mt19937 gen(rd());  uniform\_int\_distribution<> dis(a, b);  MPI\_Init(&argc, &argv);  MPI\_Request request;  MPI\_Status status;  int rank, size;  MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);  //каждый процесс генерирует свою строку массива  for (int j = 0; j < M; j++)  {  array[rank][j] = dis(gen);  cout << array[rank][j] << " ";  }  cout << endl;  //каждый процесс передает остальным элементы кроме элементов диагонали [0][0] и тд.  for (int j = 0; j < M; j++)  {  if (rank != j)  {  MPI\_Send(&array[rank][j], 1, MPI\_INT, j, 0, MPI\_COMM\_WORLD);  cout << "process: " << rank << " send number to process : " << j << ": " << array[rank][j] << endl;  }  }  //каждый процесс принимает элемент кроме элементов диагонали  for (int j = 0; j < M; j++)  {  if (rank != j)  {  MPI\_Recv(&array[rank][j], 1, MPI\_INT, j, 0, MPI\_COMM\_WORLD, &status);  cout << "process: " << rank << " recieve number: " << array[rank][j] << endl;  }  }  cout << endl;  MPI\_Finalize();  for (int j = 0; j < N; j++)  {  cout << array[rank][j] << " ";  }  cout << endl;  return 0;  } |

Изображение выглядит как текст

Автоматически созданное описание