**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Параллельные алгоритмы»**

Тема: Виртуальные топологии

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Колованов Р.А. |
| Преподаватель |  | Татаринов Ю.С. |

Санкт-Петербург

2021

**Цель работы.**

Написание программы, использующую виртуальные топологии библиотеки MPI.

**Формулировка задания.**

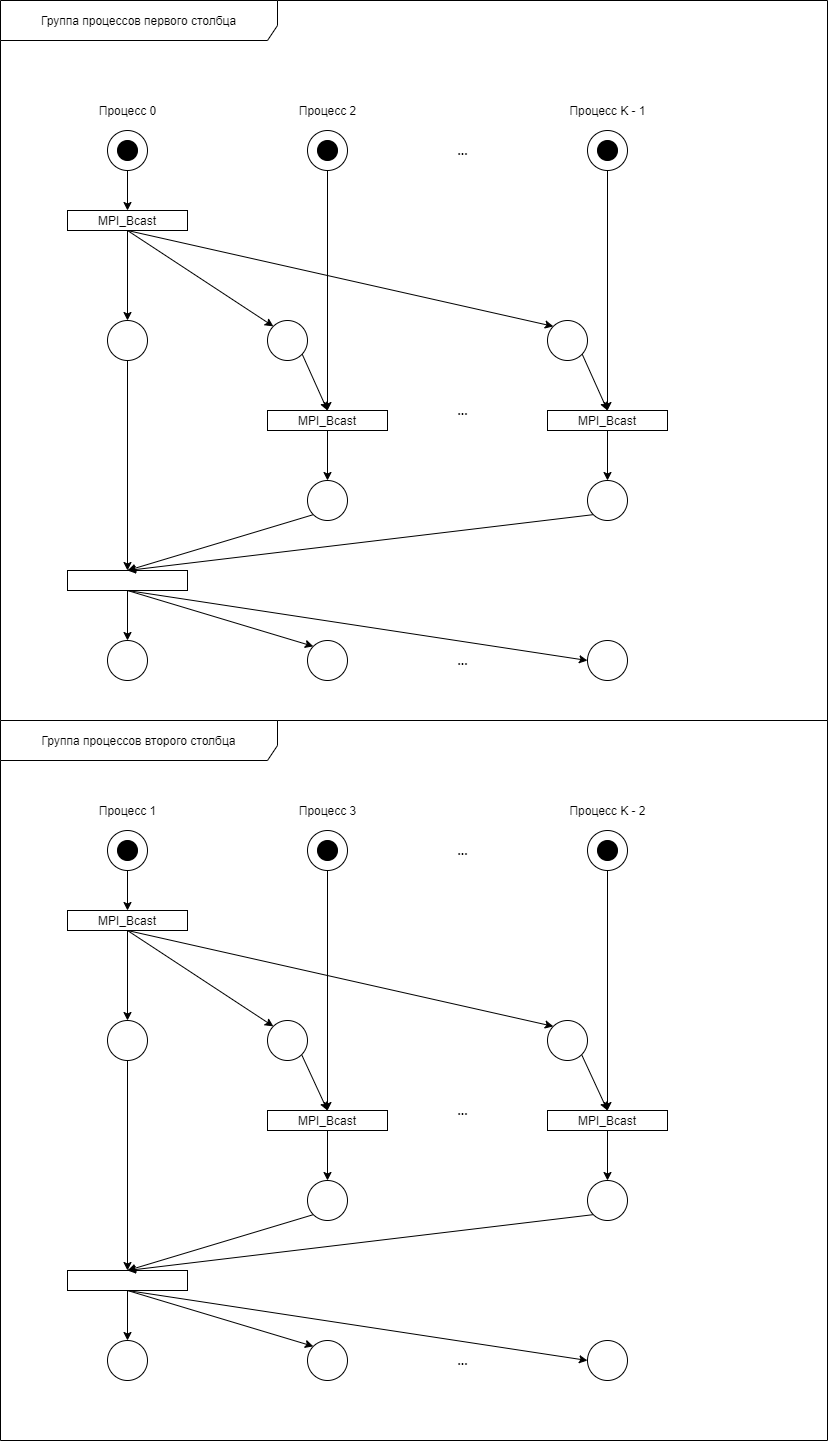
*Вариант 4.*

Число процессов К является четным: K = 2N, N > 1 В процессах 0 и 1 дано по одному вещественному числу A. Определить для всех процессов декартову топологию в виде матрицы размера N×2, после чего, используя функцию MPI\_Cart\_sub, расщепить матрицу процессов на два одномерных столбца (при этом процессы 0 и 1 будут главными процессами в полученных столбцах). Используя одну коллективную операцию пересылки данных, переслать число A из главного процесса каждого столбца во все процессы этого же столбца и вывести полученное число в каждом процессе (включая процессы 0 и 1).

**Краткое описание алгоритма.**

Для начала для процессов коммутатора MPI\_COMM\_WORLD была определена декартова виртуальная топология в виде матрицы N на 2. Далее полученная матрица была расщеплена при помощи функции MPI\_Cart\_sub на два столбца размера N, в которых главные процессы – это 0 и 1 процесс. Для процессов 0 и 1 было задано вещественное число 5 и 2.5 соответственно. Далее при помощи MPI\_Bcast производится коллективная операция отправки из главного процесса каждого столбца во все процессы этого же столбца. В конце каждый процесс выводит полученное число на экран.

**Формальное описание алгоритма.**



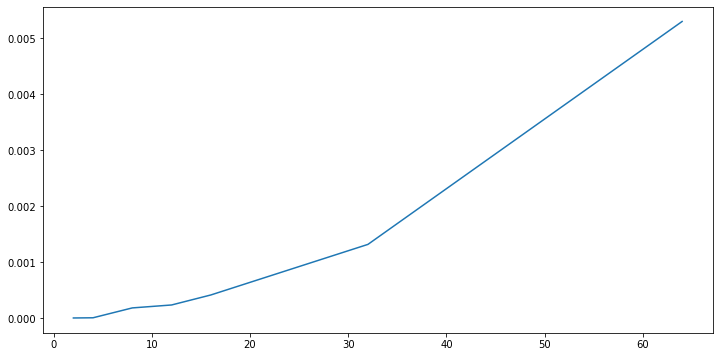
**Листинг программы.**

|  |
| --- |
| **Листинг 1. Код программы.**  #include <iostream>  #include <mpi.h>  int main(int argc, char\*\* argv) {  int processNumber, processRank;  MPI\_Status status;  MPI\_Comm matrixComm;  MPI\_Init(&argc, &argv);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &processNumber);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &processRank);  if (processNumber % 2 == 1) {  if (processRank == 0) {  std::cout << "The number of processes must be even.\n";  }  MPI\_Finalize();  return 0;  }  int dimensions[2] = { processNumber / 2 , 2 };  int periods[2] = { 0, 0 };  MPI\_Cart\_create(MPI\_COMM\_WORLD, 2, dimensions, periods, 1, &matrixComm);  MPI\_Comm rowComm;  int subdimensions[2] = { true, false };  MPI\_Cart\_sub(matrixComm, subdimensions, &rowComm);    double data = 0.0;  if (processRank == 0) {  data = 5;  } else if (processRank == 1) {  data = 2.5;  }  double startTime = MPI\_Wtime();  MPI\_Bcast(&data, 1, MPI\_DOUBLE, 0, rowComm);  double elapsedTime = MPI\_Wtime() - startTime;  std::cout << "Process: " << processRank << ", data: " << data << "\n";  double maxTime;  MPI\_Reduce(&elapsedTime, &maxTime, 1, MPI\_DOUBLE, MPI\_MAX, 0, MPI\_COMM\_WORLD);  if (processRank == 0) {  std::cout << "Elapsed time: " << maxTime << " seconds\n";  }  MPI\_Finalize();  return 0;  } |

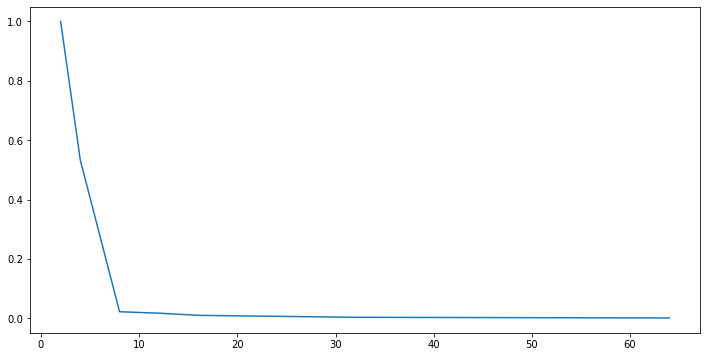
**Результаты работы программы на различном количестве процессов.**

|  |  |  |
| --- | --- | --- |
| № п/п | Количество процессоров | Результаты работы программы |
| 1. | 2 | Process: 1, data: 2.5  Process: 0, data: 5  Elapsed time: 4e-06 seconds |
| 2. | 4 | Process: 1, data: 2.5  Process: 3, data: 2.5  Process: 0, data: 5  Process: 2, data: 5  Elapsed time: 7.5e-06 seconds |
| 3. | 8 | Process: 2, data: 5  Process: 3, data: 2.5  Process: 7, data: 2.5  Process: 6, data: 5  Process: 4, data: 5  Process: 5, data: 2.5  Process: 1, data: 2.5  Process: 0, data: 5  Elapsed time: 0.0001838 seconds |
| 5. | 12 | Process: 3, data: 2.5  Process: 10, data: 5  Process: 11, data: 2.5  Process: 5, data: 2.5  Process: 0, data: 5  Process: 4, data: 5  Process: 6, data: 5  Process: 8, data: 5  Process: 7, data: 2.5  Process: 1, data: 2.5  Process: 2, data: 5  Process: 9, data: 2.5  Elapsed time: 0.0002362 seconds |
| 6. | 16 | Process: 2, data: 5  Process: 13, data: 2.5  Process: 10, data: 5  Process: 8, data: 5  Process: 12, data: 5  Process: 14, data: 5  Process: 9, data: 2.5  Process: 1, data: 2.5  Process: 6, data: 5  Process: 7, data: 2.5  Process: 11, data: 2.5  Process: 5, data: 2.5  Process: 15, data: 2.5  Process: 3, data: 2.5  Process: 4, data: 5  Process: 0, data: 5  Elapsed time: 0.0004151 seconds |
| 7. | 32 | Process: 3, data: 2.5  Process: 7, data: 2.5  Process: 30, data: 5  Process: 15, data: 2.5  Process: 9, data: 2.5  Process: 8, data: 5  Process: 20, data: 5  Process: 0, data: 5  Process: 18, data: 5  Process: 19, data: 2.5  Process: 25, data: 2.5  Process: 12, data: 5  Process: 13, data: 2.5  Process: 14, data: 5  Process: 2, data: 5  Process: 1, data: 2.5  Process: 26, data: 5  Process: 21, data: 2.5  Process: 27, data: 2.5  Process: 6, data: 5  Process: 11, data: 2.5  Process: 4, data: 5  Process: 5, data: 2.5  Process: 22, data: 5  Process: 10, data: 5  Process: 31, data: 2.5  Process: 16, data: 5  Process: 17, data: 2.5  Process: 28, data: 5  Process: 29, data: 2.5  Process: 24, data: 5  Process: 23, data: 2.5  Elapsed time: 0.0013187 seconds |
| 8. | 64 | Process: 5, data: 2.5  Process: 47, data: 2.5  Process: 27, data: 2.5  Process: 11, data: 2.5  Process: 24, data: 5  Process: 0, data: 5  Process: 18, data: 5  Process: 63, data: 2.5  Process: 39, data: 2.5  Process: 51, data: 2.5  Process: 8, data: 5  Process: 26, data: 5  Process: 23, data: 2.5  Process: 20, data: 5  Process: 41, data: 2.5  Process: 4, data: 5  Process: 25, data: 2.5  Process: 44, data: 5  Process: 58, data: 5  Process: 46, data: 5  Process: 10, data: 5  Process: 19, data: 2.5  Process: 53, data: 2.5  Process: 40, data: 5  Process: 28, data: 5  Process: 45, data: 2.5  Process: 15, data: 2.5  Process: 48, data: 5  Process: 43, data: 2.5  Process: 52, data: 5  Process: 16, data: 5  Process: 3, data: 2.5  Process: 30, data: 5  Process: 12, data: 5  Process: 13, data: 2.5  Process: 14, data: 5  Process: 29, data: 2.5  Process: 62, data: 5  Process: 6, data: 5  Process: 1, data: 2.5  Process: 50, data: 5  Process: 35, data: 2.5  Process: 31, data: 2.5  Process: 9, data: 2.5  Process: 7, data: 2.5  Process: 21, data: 2.5  Process: 42, data: 5  Process: 38, data: 5  Process: 32, data: 5  Process: 59, data: 2.5  Process: 22, data: 5  Process: 34, data: 5  Process: 17, data: 2.5  Process: 37, data: 2.5  Process: 54, data: 5  Process: 57, data: 2.5  Process: 36, data: 5  Process: 60, data: 5  Process: 33, data: 2.5  Process: 55, data: 2.5  Process: 2, data: 5  Process: 61, data: 2.5  Process: 49, data: 2.5  Process: 56, data: 5  Elapsed time: 0.0053017 seconds |

**График зависимости времени выполнения программы от числа процессов.**



**График ускорения.**



**Выводы по работе.**

Была написана программа, осуществляющая отправку вещественных чисел из главных процессов столбца остальным процессам соответствующего столбца. Было выполнено измерение времени работы с разным количеством процессов. С ростом числа процессов будет расти количество отправлений вещественного числа. Отсюда при увеличении количества процессов можно предположить, что время работы программы увеличивается линейно.