**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Параллельные алгоритмы»**

Тема: Использование функций обмена данными «точка-точка» в библиотеке MPI.

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

Санкт-Петербург

2021

**Цель работы.**

Написание программы, использующую функции обмена «точка-точка» библиотеки MPI.

**Формулировка задания.**

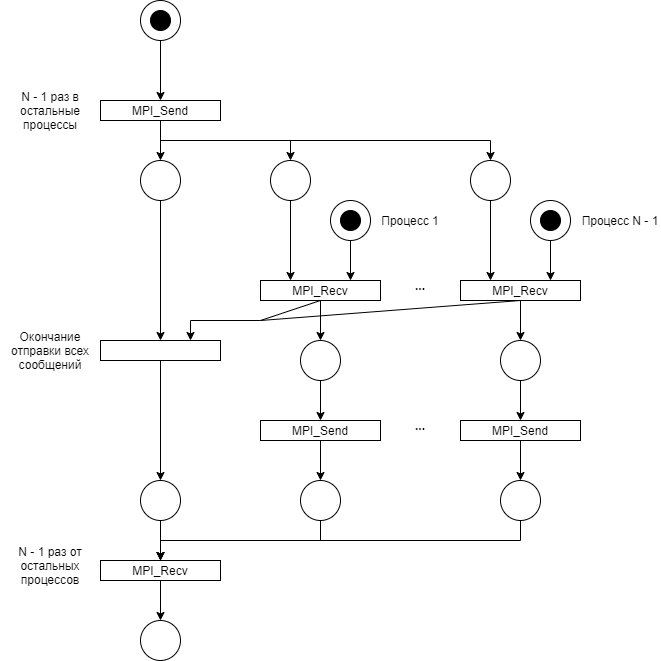
Процесс 0 генерирует массив и раздает его другим процессам для обработки (например, поиска нулевых элементов), после чего собирает результат.

В качестве обработки был выбран подсчет количество нулей в массиве.

**Краткое описание алгоритма.**

В самом начале процесс 0 генерирует массив заданного размера со случайными числами (в диапазоне от -5 до 5), после чего он делится на N – 1 равных частей, где N – количество процессов. Части массива отправляются в соответствующие процессы, которые их принимают, осуществляют подсчет количества нулей в принятой части массива и после чего отправляют результат процессу 0. Процесс 0 принимает от остальных процессов результаты, суммирует их и выводит наи экран. Если размер массиве делится на N – 1 не нацело, то остаточная часть массива обрабатывается процессом 0.

**Формальное описание алгоритма.**



**Листинг программы.**

|  |
| --- |
| **Листинг 1. Код программы.**  #include <iostream>  #include <mpi.h>  int main(int argc, char\*\* argv) {  const int dataSize = 100000000;  int processNumber, processRank;  MPI\_Status status;  srand(time(nullptr));  MPI\_Init(&argc, &argv);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &processNumber);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &processRank);  int dataBlockSize, remainDataSize;  if (processNumber != 1) {  dataBlockSize = dataSize / (processNumber - 1);  remainDataSize = dataSize % (processNumber - 1);  } else {  dataBlockSize = 0;  remainDataSize = dataSize;  }  if (processRank == 0) {  int count = 0;  int\* data = new int[dataSize];  for (int i = 0; i < dataSize; ++i) {  data[i] = rand() % 11 - 5;  }  double startTime = MPI\_Wtime();  for (int i = 1; i < processNumber; ++i) {  MPI\_Send(data + (i - 1) \* dataBlockSize, dataBlockSize, MPI\_INT, i, 0, MPI\_COMM\_WORLD);  }  for (int i = dataSize - 1; dataSize - 1 - remainDataSize < i; --i) {  if (data[i] == 0) {  ++count;  }  }  for (int result = 0, i = 1; i < processNumber; ++i) {  MPI\_Recv(&result, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD, &status);  count += result;  }  double deltaTime = MPI\_Wtime() - startTime;  delete[] data;  std::cout << "Number of '0' in array: " << count << "\n";  std::cout << "Elapsed time: " << deltaTime << "\n";  }  else {  int count = 0;  int\* data = new int[dataBlockSize];  MPI\_Recv(data, dataBlockSize, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, &status);  for (int i = 0; i < dataBlockSize; ++i) {  if (data[i] == 0) {  ++count;  }  }  MPI\_Send(&count, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);  }  MPI\_Finalize();  return 0;  } |

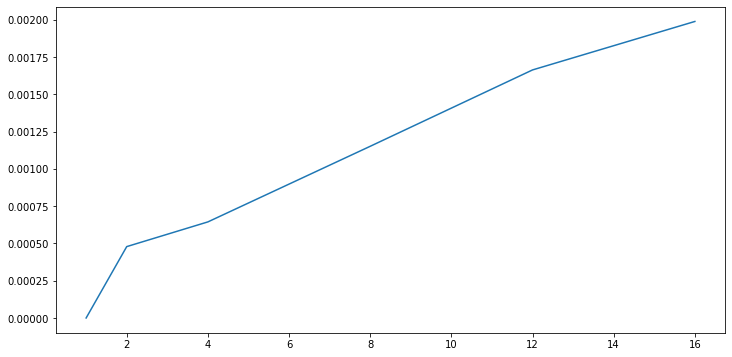
**Результаты работы программы на различном количестве процессов.**

Для N = 100000000:

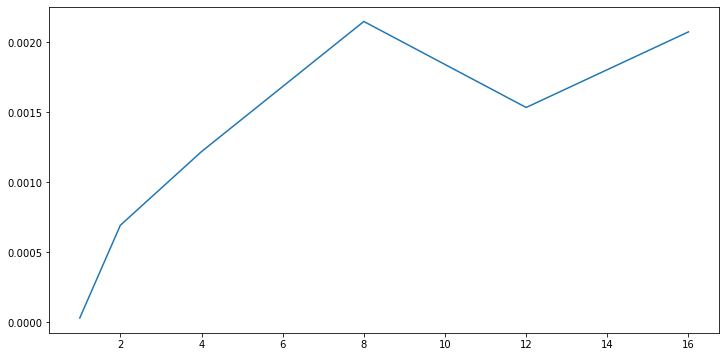
|  |  |  |
| --- | --- | --- |
| № п/п | Количество процессоров | Результаты работы программы |
| 1. | 1 | Number of '0' in array: 10  Elapsed time: 0.318001 |
| 2. | 2 | Number of '0' in array: 7  Elapsed time: 0.392543 |
| 3. | 4 | Number of '0' in array: 7  Elapsed time: 0.181756 |
| 5. | 8 | Number of '0' in array: 15  Elapsed time: 0.131727 |
| 6. | 12 | Number of '0' in array: 5  Elapsed time: 0.114662 |
| 7. | 16 | Number of '0' in array: 12  Elapsed time: 0.109926 |

**График зависимости времени выполнения программы от числа процессов для разных длин пересылаемых сообщений.**

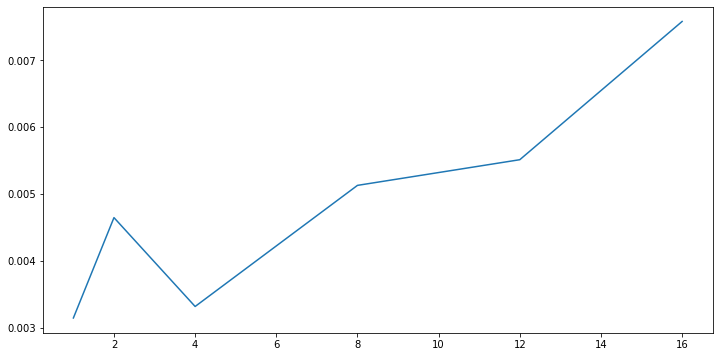
Длина массива равна 100:



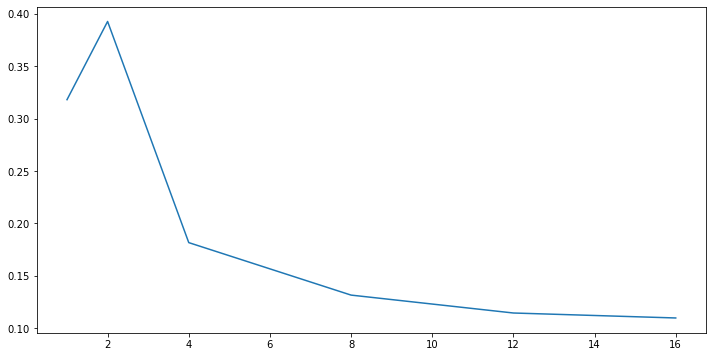
Длина массива равна 10000:



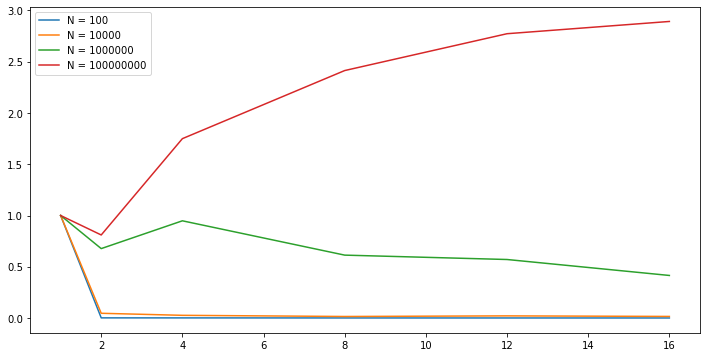
Длина массива равна 1000000:



Длина массива равна 100000000:



**График ускорения.**



**Выводы по работе.**

Была написана программа, осуществляющая параллельный подсчет количество нулей в массиве случайных чисел. Было выполнено измерение времени обработки при различных размерах сообщения с разным количеством процессов.

Исходя из полученного графика ускорения видно, что при малых размерах массива отправка и прием частей массива в остальные процессы только замедляет работу программы (быстрее просто пройтись по линейному участку памяти в одном процессе, чем осуществлять отправку частей массива в другие процессы), но при достаточно больших размерах массива параллельная обработка происходит быстрее, чем обработка в одном процессе.