**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: Коллективные операции

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

Санкт-Петербург

2021

**Цель работы.**

Написание программы, использующую коллективные функции обмена библиотеки MPI.

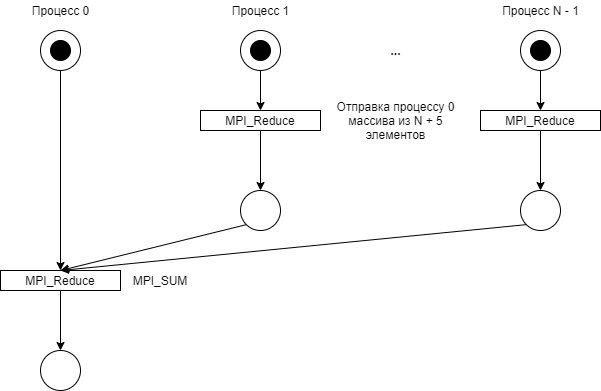
**Формулировка задания.**

В каждом процессе дан набор из K + 5 целых чисел, где K — количество процессов. Используя функцию MPI\_Reduce для операции MPI\_SUM, просуммировать элементы данных наборов с одним и тем же порядковым номером и вывести полученные суммы в главном процессе.

**Краткое описание алгоритма.**

Для начала в каждом процессе генерируется массив из N + 5 случайных чисел. Далее при помощи функции MPI\_Reduce с операцией MPI\_SUM происходит отправка сгенерированного массива чисел от каждого процесса процессу 0 с последующим суммированием элементов массива с одинаковым индексом. После получения данных процесс 0 печатает N + 5 чисел, полученных в результате суммирования элементов массивов с одинаковым индексом.

**Формальное описание алгоритма.**



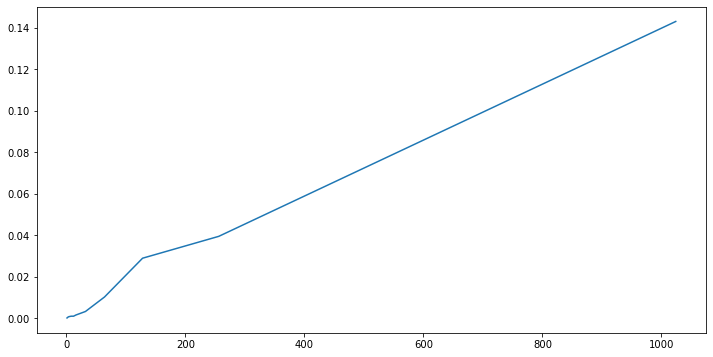
**Листинг программы.**

|  |
| --- |
| **Листинг 1. Код программы.**  #include <iostream>  #include <mpi.h>  int main(int argc, char\*\* argv) {  int processNumber, processRank;  MPI\_Init(&argc, &argv);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &processNumber);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &processRank);  srand(time(nullptr) + static\_cast<time\_t>(1000) \* processRank);  int\* sendBuffer = sendBuffer = new int[processNumber + 5];  int\* receiveBuffer = new int[processNumber + 5];  for (int i = 0; i < processNumber + 5; ++i) {  sendBuffer[i] = rand() % 11;  }  double startTime = MPI\_Wtime();  MPI\_Reduce(sendBuffer, receiveBuffer, processNumber + 5, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);  double elapsedTime = MPI\_Wtime() - startTime;  if (processRank == 0) {  for (int i = 0; i < processNumber + 5; ++i) {  std::cout << "[" << i + 1 << "] " << receiveBuffer[i] << "\n";  }  std::cout << "Elapsed time: " << elapsedTime << " sec\n";  }  delete[] sendBuffer;  delete[] receiveBuffer;  MPI\_Finalize();  return 0;  } |

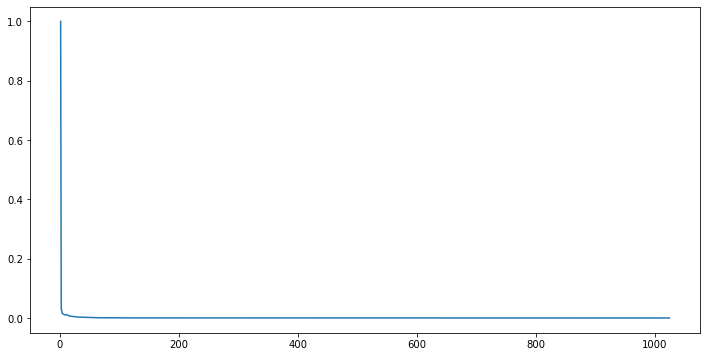
**Результаты работы программы на различном количестве процессов.**

|  |  |  |
| --- | --- | --- |
| № п/п | Количество процессоров | Результаты работы программы |
| 1. | 1 | [1] 2  [2] 8  [3] 6  [4] 1  [5] 10  [6] 8  Elapsed time: 9.30019e-06 sec |
| 2. | 2 | [1] 15  [2] 14  [3] 4  [4] 10  [5] 10  [6] 6  [7] 8  Elapsed time: 0.0002987 sec |
| 3. | 4 | [1] 13  [2] 29  [3] 20  [4] 20  [5] 16  [6] 19  [7] 21  [8] 19  [9] 21  Elapsed time: 0.0006283 sec |
| 5. | 8 | [1] 38  [2] 54  [3] 40  [4] 53  [5] 45  [6] 42  [7] 40  [8] 4  [9] 27  [10] 38  [11] 37  [12] 50  [13] 44  Elapsed time: 0.0008678 sec |
| 6. | 12 | [1] 51  [2] 53  [3] 57  [4] 66  [5] 53  [6] 61  [7] 63  [8] 37  [9] 58  [10] 52  [11] 66  [12] 68  [13] 73  [14] 64  [15] 73  [16] 62  [17] 59  Elapsed time: 0.0011785 sec |
| 7. | 16 | [1] 76  [2] 70  [3] 96  [4] 78  [5] 67  [6] 83  [7] 81  [8] 104  [9] 89  [10] 82  [11] 78  [12] 74  [13] 97  [14] 80  [15] 85  [16] 81  [17] 83  [18] 76  [19] 93  [20] 81  [21] 86  Elapsed time: 0.0014459 sec |

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



**График ускорения.**



**Выводы по работе.**

Была написана программа, осуществляющая суммирование элементов с одинаковыми индексами в массивах случайных чисел, генерируемых каждым процессом. Было выполнено измерение времени работы с разным количеством процессов. С ростом числа процессов будет расти и количество передаваемых массивов, и их размер. Отсюда при увеличении количества процессов время работы программы тоже увеличивается.