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: Коллективные операции

Студент гр. 9383 _____ Лапина А.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург

2021

Цель работы.

Написать программу пересылки данных из главного процесса другим, используя функцию **MPI_Scatter**.

Формулировка задания.

Вариант 3.

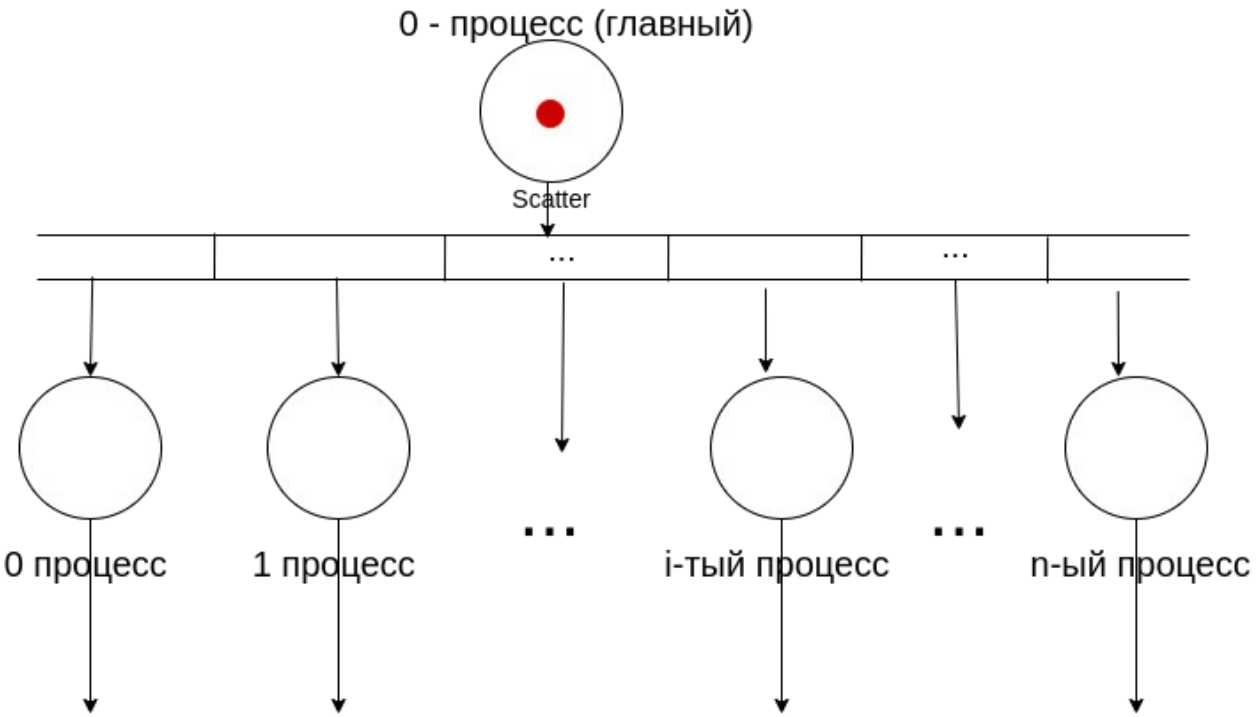
В главном процессе дан набор из 3*K* чисел, где *K* — количество процессов. Используя функцию **MPI_Scatter**, переслать по 3 числа в каждый процесс (включая главный) и вывести в каждом процессе полученные числа.

Краткое описание выбранного алгоритма решения задачи.

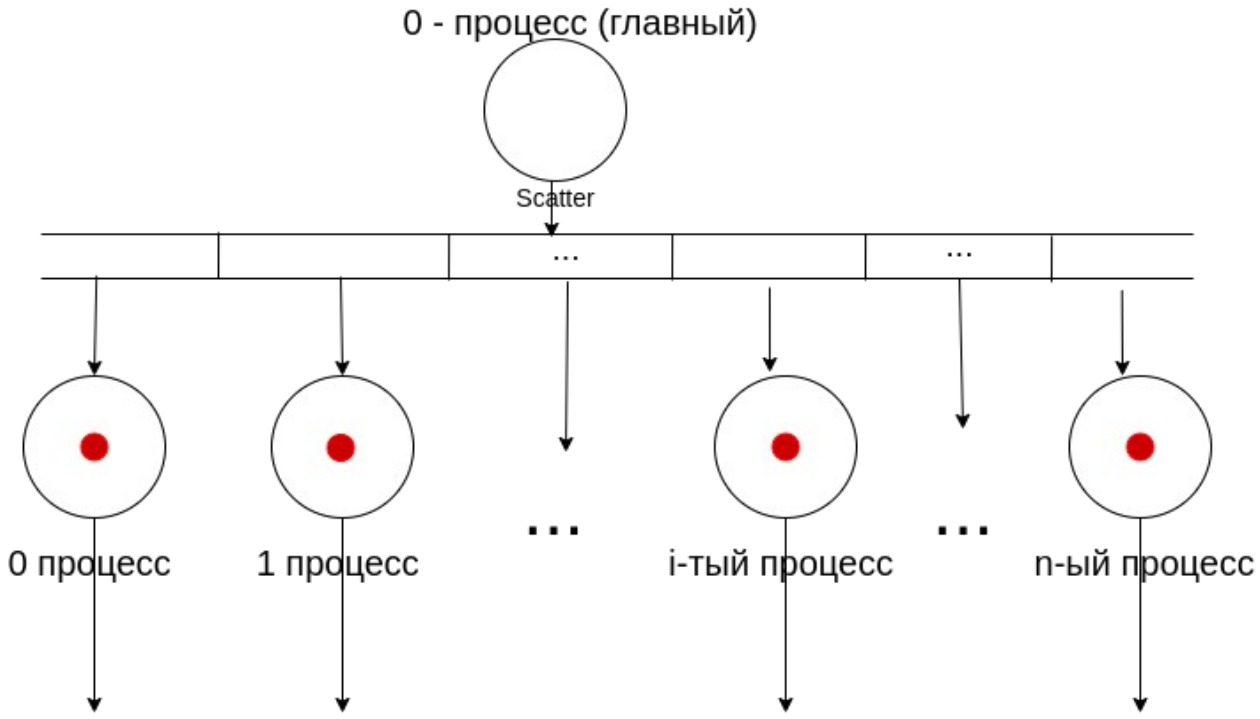
Главным процессом является нулевой процесс. Создаем переменную Size размером 3*{количество процессов}, измеряем время до начала выполнения программы, записываем его в переменную Start. Затем в нулевом процессе заполняем массив array рандомными числами (до 100). С помощью функции MPI_Scatter() рассылаем по 3 числа из сгенерированного массива array, в новый массив SendArr каждому процессу, включая нулевой, после в каждом массиве выводим пересланные данные. Измеряем время и записываем его в Finish. Выводим разность Finish и Start - общее время работы программы.

Формальное описание алгоритма с использованием аппарата Сетей Петри для n процессов:

Шаг 1 — формируем массив в главном процессе - 0 и рассылаем его по частям всем n процессам, включая нулевой:



Следующий шаг:



Листинг программы.

```
#include <stdio.h>
#include "stdlib.h"
#include <mpi.h>
#include "time.h"

void main( int argc, char *argv[] ) {
   srand(time(NULL));
   int i, ProcRank, ProcNum, RecvRank;
   MPI_Init(&argc, &argv);
   MPI_Status Status;
   MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
   MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
   int Size = 3 * ProcNum;
   int array[Size];
   double Start = MPI_Wtime();
   int SendArr[3];
   if (ProcRank == 0){
      for (i = 0; i < Size; i++){
         array[i] = rand() % 100;
         printf("%d  ", array[i]);
         if (i == Size - 1)
            printf("\n");
      }
   }
   MPI_Scatter(array, 3, MPI_INT, SendArr, 3, MPI_INT, 0, MPI_COMM_WORLD);
   printf("Array by process %d: ", ProcRank);
   for(i = 0; i<3; i++){
      printf("%d ", SendArr[i]);
   }
   printf("\n");
   MPI_Finalize();
}
```

Результаты работы программы

1) для 4 процессов

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 4 ./l3.x
48  34  47  2  73  6  63  63  39  83  50  35
Array by process 0: 48 34 47
Array by process 2: 63 63 39
Array by process 1: 2 73 6
Array by process 3: 83 50 35
Time: 0.000053
```

2) для 6 процессов

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 6 ./l3.x
96  81  56  73  22  66  68  97  90  34  90  38  53  77  85  59  92  49
Array by process 0: 96 81 56
Array by process 2: 68 97 90
Array by process 3: 34 90 38
Array by process 1: 73 22 66
Array by process 4: 53 77 85
Array by process 5: 59 92 49
Time: 0.000178
```

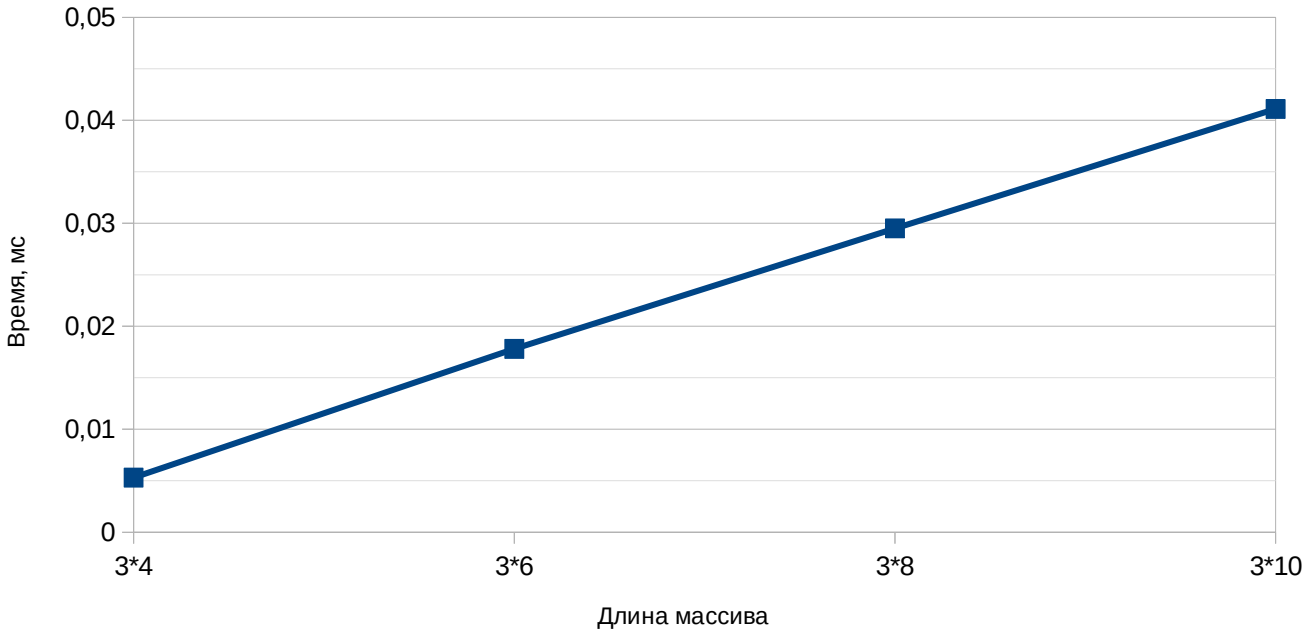
3) для 8 процессов

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 8 ./l3.x
40  46  60  62  6  64  27  49  72  50  5  97  64  85  61  19  61  91  14  98  6  40  50  23
Array by process 0: 40 46 60
Array by process 1: 62 6 64
Array by process 2: 27 49 72
Array by process 3: 50 5 97
Array by process 4: 64 85 61
Array by process 5: 19 61 91
Array by process 6: 14 98 6
Array by process 7: 40 50 23
Time: 0.000295
```

4) для 10 процессов

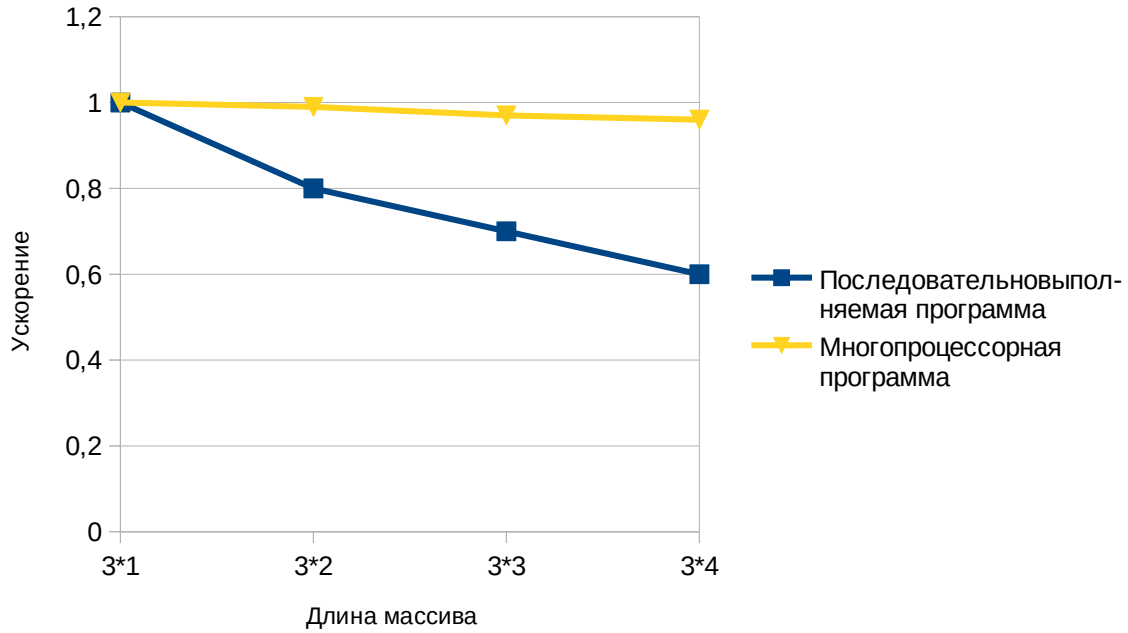
```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 10 ./l3.x
90  86  96  96  8  61  67  98  11  45  6  27  47  89  51  69  63  35  53  29  47  87  99  14  83  29  10  59  4  23
Array by process 0: 90 86 96
Array by process 1: 96 8 61
Array by process 2: 67 98 11
Array by process 3: 45 6 27
Array by process 4: 47 89 51
Array by process 5: 69 63 35
Array by process 6: 53 29 47
Array by process 7: 87 99 14
Array by process 8: 83 29 10
Array by process 9: 59 4 23
Time: 0.000411
```

График зависимости времени выполнения программы от разных длин массива.



При увеличении длины сообщения время выполнения программы возрастает. Это происходит потому что главный массив отправляет по 3 числа другим процессам, поэтому логично, что при увеличении количества элементов в массиве время незначительно увеличивается.

График ускорения (эффективности):



Для однопроцессорной программы ускорение убывает быстрее, чем с программой использующей несколько процессов. Это происходит потому что при использовании нескольких процессов программа распараллеливается и процессы выполняются одновременно, в отличии от однопроцессорной программы, где все действия, такие как: отправка частей массива и их вывод выполняются последовательно.

(Длина массива = 3*{количество процессов} по заданию).

Вывод.

Была написана программа пересылки данных из главного процесса другим, используя функцию **MPI_Scatter**. а также проанализирована работа написанной многопроцессной программы.