МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №01

по дисциплине «Параллельные алгоритмы»

Тема: Обмен сообщениями чётных и нечётных процессов

Студент гр. 9383 _____ Лапина А.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург

2021

Цель работы.

Написать программу обмена сообщениями чётных и нечётных процессов. Проанализировать зависимость времени обмена от длины сообщения.

Формулировка задания.

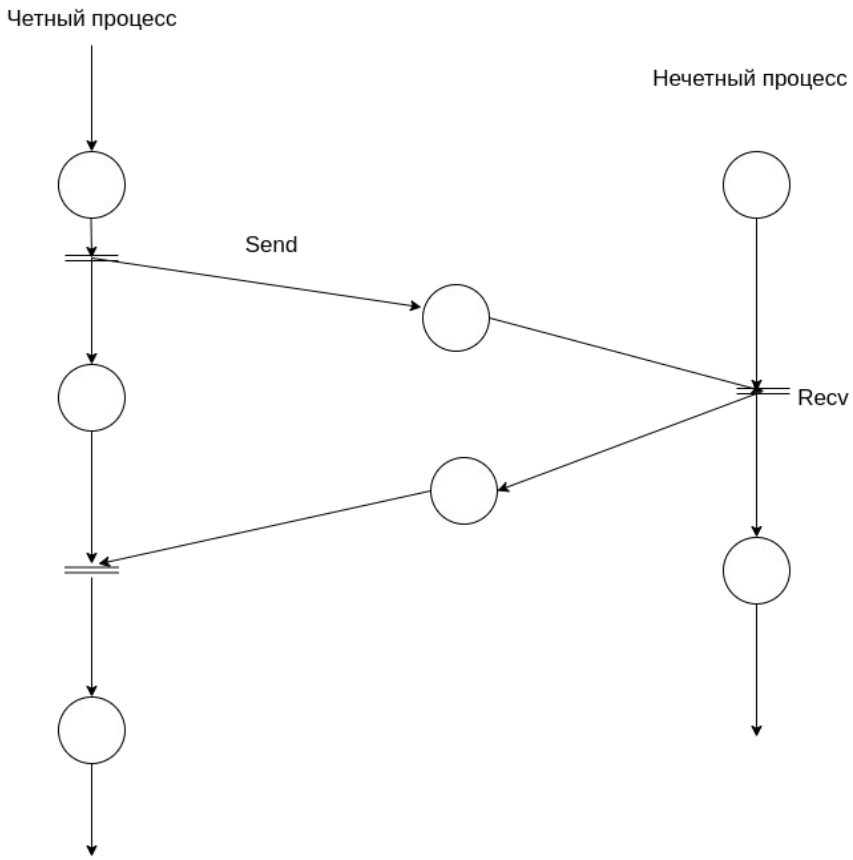
Напишите программу обмена сообщениями чётных и нечётных процессов. Замерьте время на одну итерацию обмена и определите зависимость времени обмена от длины сообщения.

Краткое описание выбранного алгоритма решения задачи.

Задаем переменные: Size - отвечает за длину сообщения , массивы размером Size SendData, RecvData для передачи и приема сообщений между процессами, заполним массив SendData номером элемента, для анализа программы будем считать среднее время, затраченное на процесс, для этого заведем переменную SumTime, которая с помощью MPI_Reduce посчитает суммарное время выполнения каждого процесса.

В начале выполнения программы измеряем время с помощью MPI_Wtime(), результат записываем в переменную Start, затем выполняем обмен сообщения между четными и нечетными процессами с помощью MPI_Recv и MPI_Send таким образом, что 0 процесс обменивается с 1, 2 с 3, 4 с 5 и тд. После выполнения работы, измеряем время и сохраняем его в переменной Finish. После считаем суммарное время выполнения процессов и делим на их количество, таким образом выводим информацию о среднем времени, затраченном на процесс.

Формальное описание алгоритма с использованием аппарата Сетей Петри:



Листинг программы.

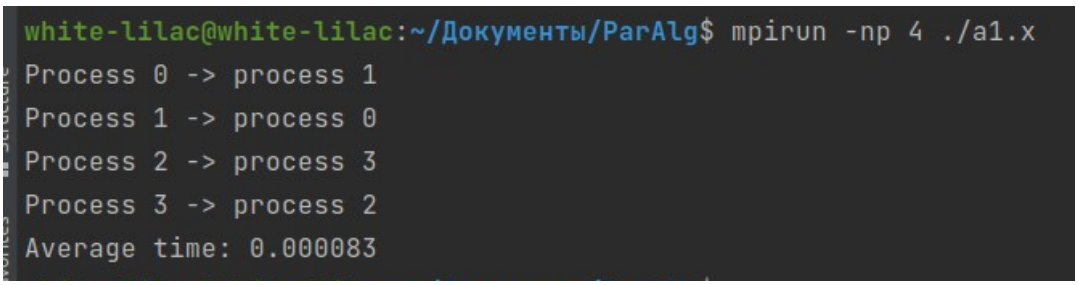
```
#include <stdio.h>

#include "mpi.h"

int main(int argc, char* argv[]){

    int Size = 100000;

    int SendData[Size];

    int RecvData[Size];

    double SumTime = 0;

    int ProcNum, ProcRank, RecvRank;

    MPI_Status Status;
```

```
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
    MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
    SendData[0] = ProcRank;
    for (int i = 1; i < Size; i++) {
        SendData[i] = i;
    }
    double Start, Finish;
    Start = MPI_Wtime();
    if (ProcRank % 2 == 0){
        if (ProcRank +1 < ProcNum) {
                        MPI_Send(&SendData, Size, MPI_INT, ProcRank + 1, 0,
MPI_COMM_WORLD);
                MPI_Recv(&RecvData, Size, MPI_INT, ProcRank + 1, MPI_ANY_TAG,
MPI_COMM_WORLD, &Status);
            printf ("Process %d -> process %d\n", ProcRank, RecvData[0]);
        }
    } else {
              MPI_Recv(&RecvData, Size, MPI_INT, ProcRank - 1, MPI_ANY_TAG,
MPI_COMM_WORLD, &Status);
        MPI_Send(&SendData, Size, MPI_INT, ProcRank - 1, 0, MPI_COMM_WORLD);
        printf ("Process %d -> process %d\n", ProcRank, RecvData[0]);
    }


    Finish = MPI_Wtime();
```

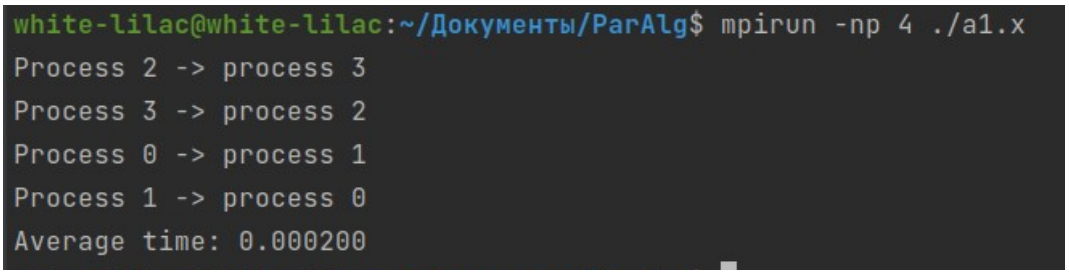
```
    double Time = Finish-Start;

            MPI_Reduce(&Time, &SumTime, 1, MPI_DOUBLE, MPI_SUM, 0,
MPI_COMM_WORLD);

    if (ProcRank == 0) {

        printf("Average time: %f\n", SumTime/ProcNum);

    }

    MPI_Finalize();

    return 0;

}
```

Результаты работы программы

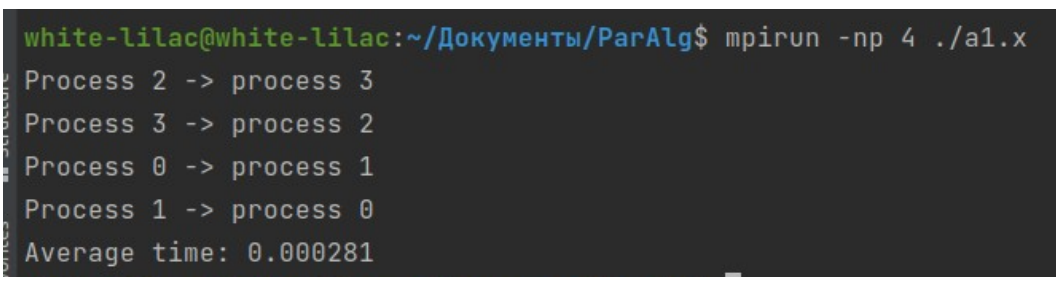
1) для 4 процессоров и длине сообщения Size = 10;



2) для 4 процессоров и длине сообщения Size = 100;



3) для 4 процессоров и длине сообщения Size = 1000;



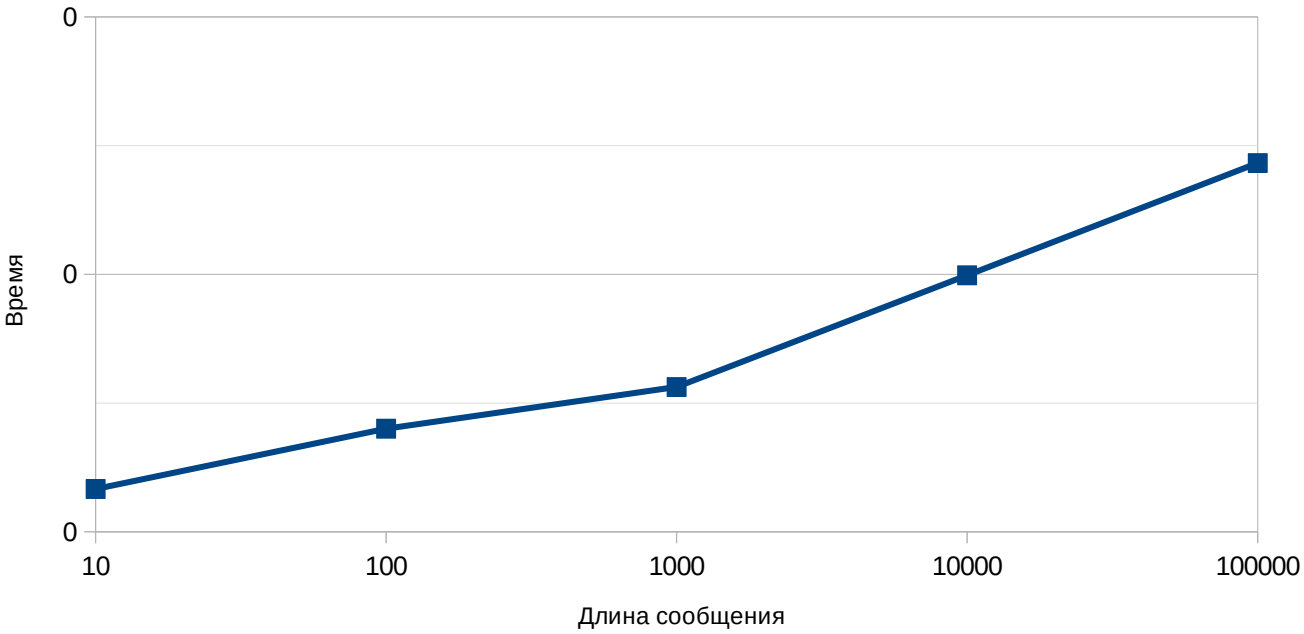
4) для 4 процессоров и длине сообщения Size = 10000;

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 4 ./a1.x
Process 0 -> process 1
Process 1 -> process 0
Process 2 -> process 3
Process 3 -> process 2
Average time: 0.000498
```

5) для 4 процессоров и длине сообщения Size = 100000;

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 4 ./a1.x
Process 2 -> process 3
Process 3 -> process 2
Process 1 -> process 0
Process 0 -> process 1
Average time: 0.000716
```

График зависимости времени выполнения программы от числа процессов равному 4 для разных длин пересылаемых сообщений.



При увеличении длины сообщения для одинакового количества процессов (равного 4) время выполнения программы незначительно увеличивается (на несколько тысячных миллисекунд).

Результаты работы программы

1) для 2 процессоров и длине сообщения Size = 100000;

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 2 ./a1.x
Process 1 -> process 0
Process 0 -> process 1
Average time: 0.000437
```

2) для 4 процессоров и длине сообщения Size = 100000;

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 4 ./a1.x
Process 0 -> process 1
Process 1 -> process 0
Process 2 -> process 3
Process 3 -> process 2
Average time: 0.000446
```

3) для 6 процессоров и длине сообщения Size = 100000;

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 6 ./a1.x
Process 2 -> process 3
Process 3 -> process 2
Process 1 -> process 0
Process 0 -> process 1
Process 4 -> process 5
Process 5 -> process 4
Average time: 0.000984
```

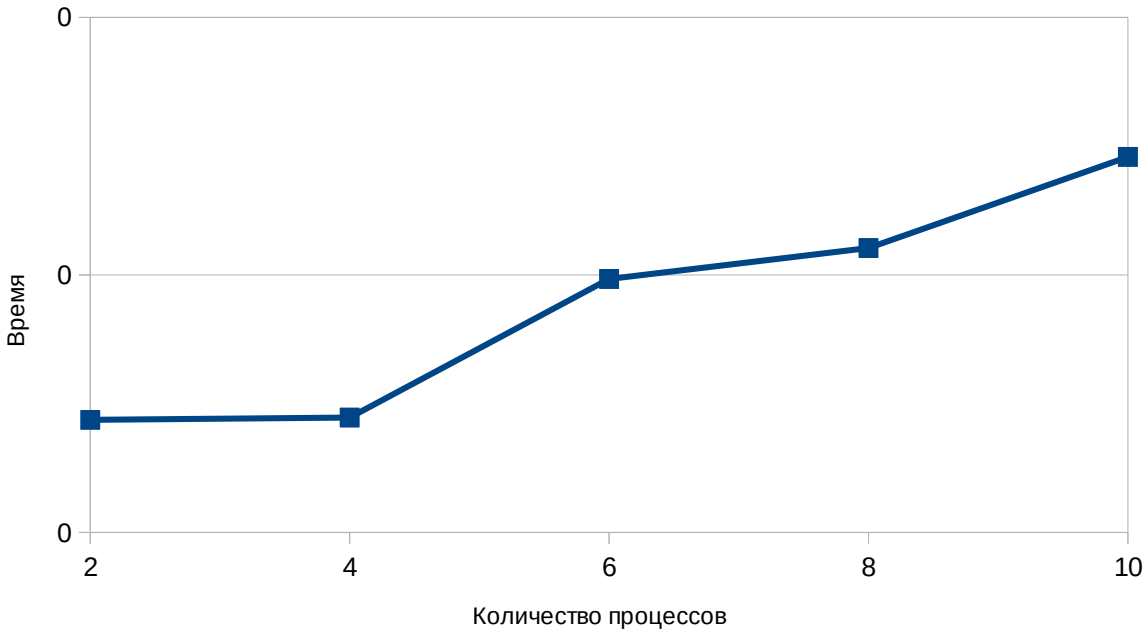
4) для 8 процессоров и длине сообщения Size = 100000;

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 8 ./a1.x
Process 6 -> process 7
Process 7 -> process 6
Process 0 -> process 1
Process 1 -> process 0
Process 3 -> process 2
Process 4 -> process 5
Process 5 -> process 4
Process 2 -> process 3
Average time: 0.001104
```

5) для 10 процессоров и длине сообщения Size = 100000;

```
white-lilac@white-lilac:~/Документы/ParAlg$ mpirun -np 10 ./a1.x
Process 1 -> process 0
Process 3 -> process 2
Process 0 -> process 1
Process 2 -> process 3
Process 7 -> process 6
Process 6 -> process 7
Process 5 -> process 4
Process 4 -> process 5
Process 9 -> process 8
Process 8 -> process 9
Average time: 0.001458
```

График ускорения (эффективности):



При увеличении количества процессов время незначительно меняется (на тысячные миллисекунд), проводился анализ для передачи сообщения довольно большого размера (Size = 100000).

Вывод.

Была написана программа обмена сообщениями чётных и нечётных процессов, а также проанализирована зависимость времени обмена от длины сообщения.