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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Системы параллельной обработки данных»

Тема: ИСПОЛЬЗОВАНИЕ КОЛЛЕКТИВНЫХ ОПЕРАЦИЙ

Студентка гр. 5303 _____ Допира В.Е.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург

2019

Формулировка задания

Задание: написать масштабируемую параллельную программу вычисления суммы элементов матрицы с использованием коллективных операций.

Описание алгоритма с использованием аппарата Сетей Петри

P_0 - начальное состояние. Нулевой процесс разделяет массив на части и передает их из P_0 остальным процессам, тем самым осуществляются переходы t_1, t_2, t_3 к состояниям P_1, P_2, P_3 соответственно. Количество переходов соответствует количеству заданных процессов минус 1. Здесь показано на трех. Далее процессы-получатели считают суммы тех частей, которые получили. Выполнение операции MPI_Reduce, срабатывает переход t_4 к состоянию P_4. Нулевой процесс выводит результат. Программа завершает выполнение.

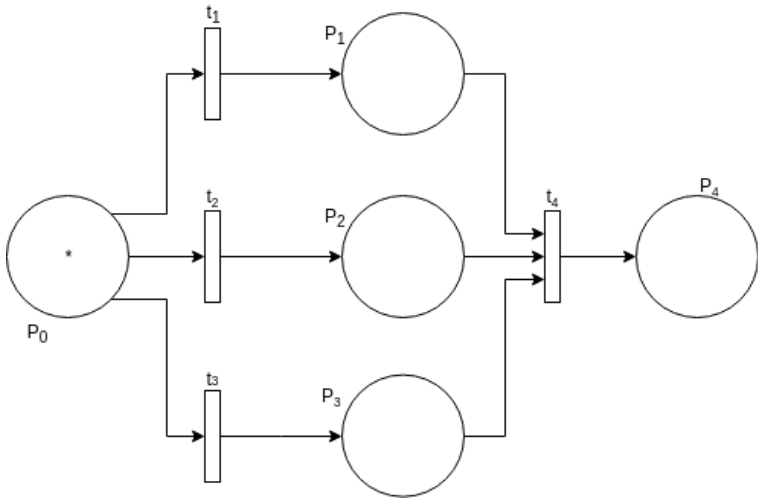

Рисунок 1 — Сеть Петри

Результаты работы программы на 1,2 …. N процессорах

```
$ mpirun 3.x
3 6 7
5 3 5
6 2 9
Result: 46
```

Вывод

В ходе выполнения лабораторной работы была написана масштабируемая программа с использованием MPI, вычисляющая сумму элементов матрицы. В программе были использованы коллективные операции.

Листинг программы

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

int main(int argc, char *argv[])
{
    int result, sum = 0;
    int ProcNum, ProcRank;

    int m = 3;
    int n = 3;
    int array[m][n];
    for(int i = 0; i < m; i++){
      for(int j = 0; j < n; j++){
          array[i][j] = rand() % 10;
            printf("%d ", array[i][j]);
      }
        printf("\n");
    }

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD,&ProcNum);
    MPI_Comm_rank(MPI_COMM_WORLD,&ProcRank);

    int step = m*n/ProcNum;
    int* pointer = (*array + ProcRank*step);
    int Num = step;
    if(ProcRank == ProcNum - 1){
        Num = Num + m*n % ProcNum;
    }
    for(int i = 0; i < Num; i++){
      sum += pointer[i];
    }

    MPI_Reduce(&sum, &result, 1, MPI_INT, MPI_SUM, 0,
MPI_COMM_WORLD);

    if (ProcRank == 0)
    {
        printf("Result: %d\n", result);
    }

    MPI_Finalize();
    return 0;
}
```