

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

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Системы параллельной обработки данных»
Тема: ИСПОЛЬЗОВАНИЕ КОЛЛЕКТИВНЫХ ОПЕРАЦИЙ

Студент гр. 5304

Лянгузов А.А.

Преподаватель

Татаринов Ю.С.

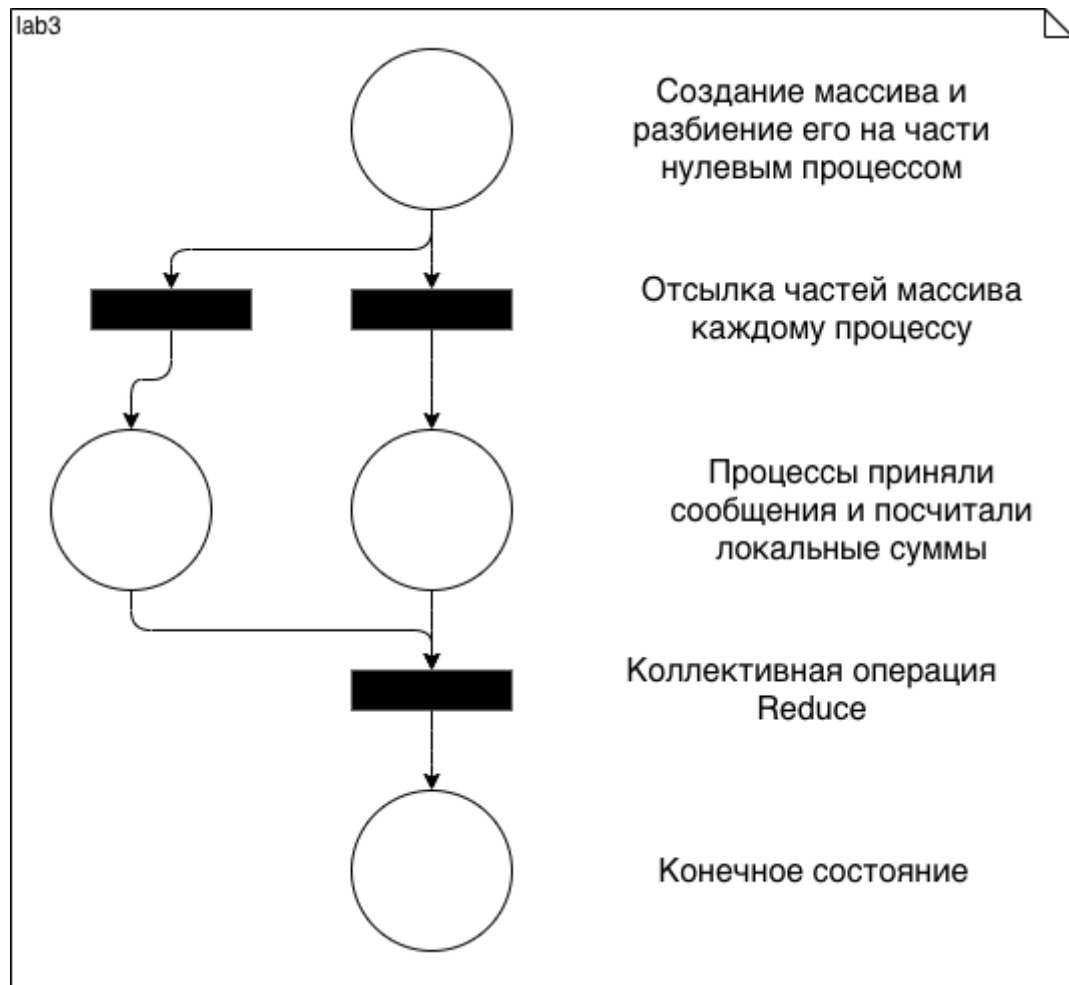
Санкт-Петербург

2019

Задание:

Написать масштабируемую параллельную программу вычисления суммы элементов динамического массива (вектора, матрицы) с использованием коллективных операций.

Сеть Петри.



Выполнение работы

За генерацию массива отвечает нулевой процесс. Так же он разбивает массив на равные части и рассылает это по процессам.

Ненулевые процессы ловят массивы и считают подсуммы.

Функция `MPI_Reduce(&Sum, &Result, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD)` складывает все суммы и отправляет нулевому процессу.

```
(base) xtail@~/Projects/LETI/Parallel/ParallelLabs$ ./assembly.sh launch lab3 1
original matrix
 7  6  2  7  0  3
 9  9  9  1  7  2
 3  6  5  5  8  1
 4  7  1  3  8  4
 8  0  4  6  0  3
Result = 138
```

Рис.1. Результат работы программы

Выводы

В ходе выполнения лабораторной работы ознакомились с коллективными операциями путем преобразования простейшей программы.

ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ

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

int main(int argc, char *argv[])
{
    int M = 5;
    int N = 6;
    int Matrix[M][N];
    for(int i = 0; i < M; i++)
    {
        for(int j = 0; j < N; j++)
        {
            Matrix[i][j] = rand() % 10;
        }
    }

    printf("original matrix\n");
    for(int i = 0; i < M; i++)
    {
        for(int j = 0; j < N; j++)
        {
            Matrix[i][j] = rand() % 10;
            printf(" %3d", Matrix[i][j]);
        }
        printf("\n");
    }

    int Result;

    int ProcNum, ProcRank;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
    MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
    int Sum = 0;
    int Step = M*N/ProcNum;
```

```
int* Pointer = (*Matrix + 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 = %3d\n", Result);
}

MPI_Finalize();
return 0;
}
```