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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Параллельные алгоритмы»

Тема: Группы процессов и коммуникаторы. Создание новых коммуникаторов.

Студент гр. 1384 _____ Усачева Д. В.

Преподаватель _____ Татаринов Ю. С.

Санкт-Петербург

2023

Цель

Цель работы заключается в изучении и применении концепций групп процессов и коммуникаторов в MPI.

Задание

Вариант №4. В каждом процессе чётного ранга (включая главный процесс) дан набор из трёх элементов — вещественных чисел. Используя новый коммуникатор и одну коллективную операцию редукции, найти минимальные значения среди элементов исходных наборов с одним и тем же порядковым номером и вывести найденные минимумы в главном процессе. Новый коммуникатор создать с помощью функции MPI_Comm_split.

Указание. При вызове функции MPI_Comm_split в процессах, которые не требуется включать в новый коммуникатор, в качестве параметра color следует указывать константу MPI_UNDEFINED.

Выполнение работы

Для выполнения поставленной задачи написана программа на языке C, код которой представлен ниже в листинге 1.

Для выполнения поставленной задачи создаются: массив размера 3 для набора элементов каждого четного процесса, результирующий массив для записи минимальных значений каждой позиции, новый коммуникатор и другие необходимые переменные. Коммуникатор включает в себя только четные процессы, для нечетных процессов номер коммуникатора, которому должен принадлежать процесс равен MPI_UNDEFINED. Далее для каждого четного процесса заполняется массив из 3 элементов.

Чтобы найти минимальные значения среди элементов исходных наборов с одним и тем же порядковым номером, используется коллективная операция редукции MPI_Reduce для нового коммуникатора. Функция вызывается для каждого процесс из данного коммуникатора, используется операция MPI_MIN для каждой позиции поочередно, минимальное число записывается в результирующий массив.

Процесс с рангом 0 получает результат и выводит его на экран. В конце своей работы каждый четный процесс освобождает созданный коммуникатор при помощи MPI_Comm_free, для нечетных процессов коммуникатор неопределён.

Ниже представлена сеть Петри основной части алгоритма (см. рис 1).

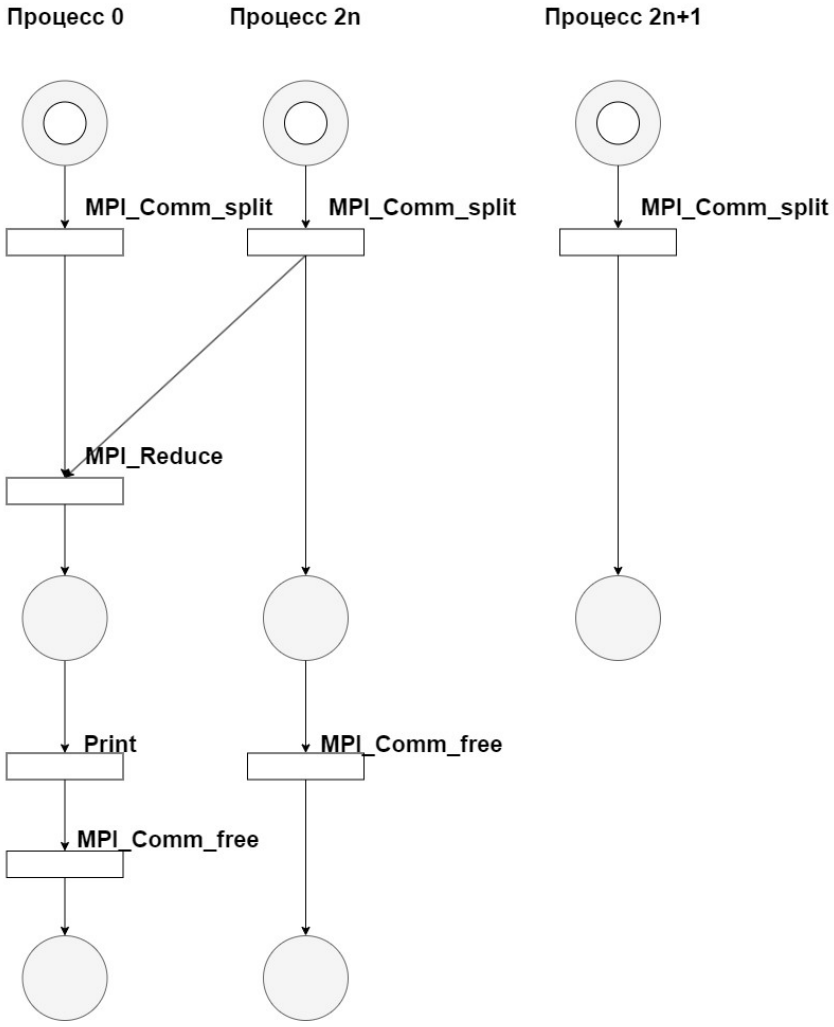

Рисунок 1 — Сеть Петри основной части алгоритма

Листинг 1 — Код программы lab4.c

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

int main(int argc, char **argv)
{
    int procNum, procRank;
    int size = 3;
    double array[size];
    double minArray[size];
    double start;
    MPI_Init(&argc, &argv);
    start = MPI_Wtime();
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);
```

```
        MPI_Comm_size(MPI_COMM_WORLD, &procNum);
        MPI_Comm newComm;
        int color = procRank % 2;
        if (color == 0)
        {
            MPI_Comm_split(MPI_COMM_WORLD, color, procRank, &newComm);
            printf("Набор элементов для процесса ранга %d: ", procRank);
            for (int i = 0; i < 3; i++)
            {
                array[i] = 1.0 * (procRank + i);
                printf("%f ", array[i]);
            }
            printf("\n");
        }
        else
        {
            color = MPI_UNDEFINED;
            MPI_Comm_split(MPI_COMM_WORLD, color, procRank, &newComm);
        }
        if (color == 0)
        {
            MPI_Reduce(&array[0], &minArray[0], size, MPI_DOUBLE, MPI_MIN, 0,
newComm);
        }
        if (procRank == 0)
        {
            printf("Минимальные значения элементов:");
            for (int i = 0; i < size; i++)
            {
                printf("%f ", minArray[i]);
            }
            printf("\nВремя работы программы: %f\n", MPI_Wtime() - start);
        }
        if (color == 0)
        {
            MPI_Comm_free(&newComm);
        }
        MPI_Finalize();

        return 0;
    }
```

Ниже представлен вывод программы lab4.c

Листинг 2 — Вывод программы lab4.c для 3, 5 и 9 процессов

```
Набор элементов для процесса ранга 0: 0.000000 1.000000 2.000000
Набор элементов для процесса ранга 2: 2.000000 3.000000 4.000000
Минимальные значения элементов:0.000000 1.000000 2.000000
Время работы программы: 0.000695

Набор элементов для процесса ранга 4: 4.000000 5.000000 6.000000
Набор элементов для процесса ранга 0: 0.000000 1.000000 2.000000
Набор элементов для процесса ранга 2: 2.000000 3.000000 4.000000
Минимальные значения элементов:0.000000 1.000000 2.000000
Время работы программы: 0.001057

Набор элементов для процесса ранга 8: 8.000000 9.000000 10.000000
Набор элементов для процесса ранга 0: 0.000000 1.000000 2.000000
Набор элементов для процесса ранга 4: 4.000000 5.000000 6.000000
Набор элементов для процесса ранга 6: 6.000000 7.000000 8.000000
Набор элементов для процесса ранга 2: 2.000000 3.000000 4.000000
Минимальные значения элементов:0.000000 1.000000 2.000000
Время работы программы: 0.001701
```

Так как в условии размер массива для четных процессов фиксированный и равен 3, рассмотрим зависимость времени работы программы от количества процессов.

Таблица 1 — Среднее время выполнения.

Количество процессов	Среднее время на выполнение(мс)
3	0,695
5	1,057
9	1,67
17	3,03
33	7,227

Ниже указаны графики зависимостей времени выполнения и ускорения (см. рисунки 2-3).

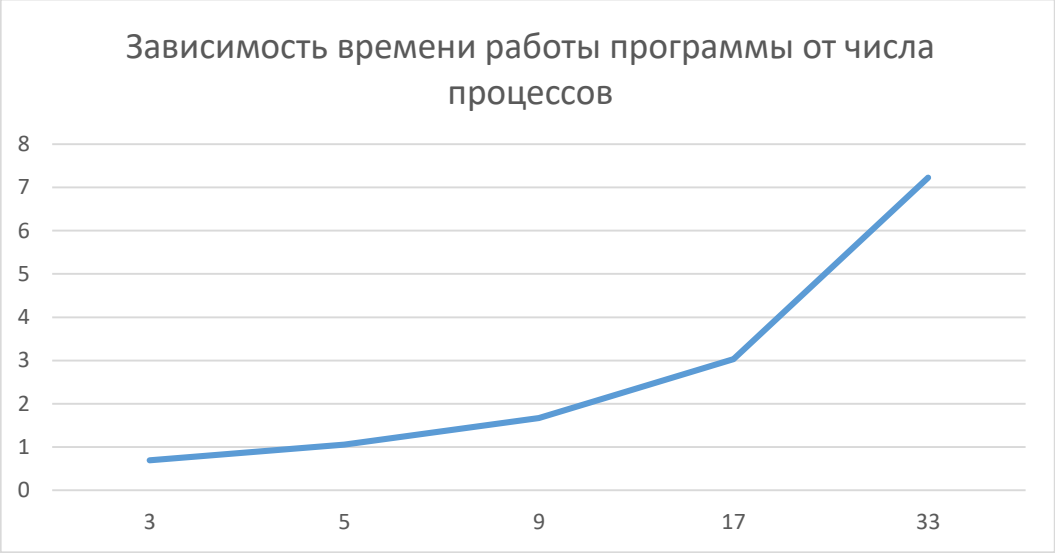

Рисунок 2 — График зависимости времени выполнения от числа процессов

Ускорение времени работы программы можно вычислить по формуле:

$$S_p(n) = T_1(n)/T_p(n)$$

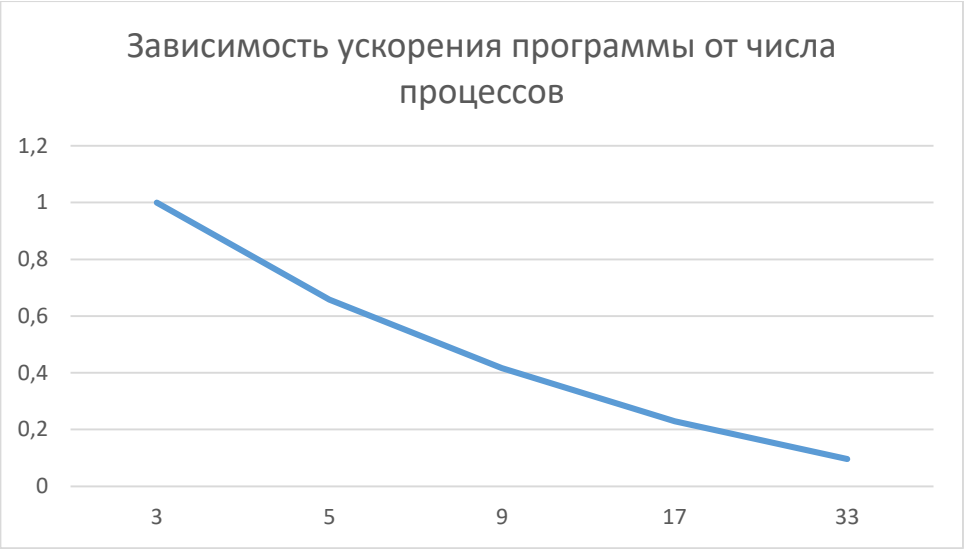

Рисунок 3 — График зависимости ускорения от числа процессов

Выводы

В ходе выполнения лабораторной работы были изучены и использованы коллективная операция MPI_Reduce и функция MPI_Comm_split. По полученным экспериментальным результатам можно сделать вывод о том, что время выполнения увеличивается при увеличении числа процессов. Это связано с тем, что при большем количестве процессов необходимо сравнить большее число элементов на каждой позиции, поэтому увеличивается время ожидания результата нулевым процессом. Так же время работы увеличивается из-за генерации большего количества массивов.