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

# ОТЧЕТ

**по лабораторной работе №6**

**по дисциплине «Системы параллельной обработки данных»**

**Тема: Виртуальные топологии**

Студент гр. 0303 _____ Калмак Д.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург
2024

**Цель работы.**

Целью работы является написание параллельной программы MPI с виртуальными топологиями, изучение зависимостей времени работы программы от числа процессов и построение сети Петри.

**Задание.**

1. Число процессов K является четным: K = 2N, N > 1. В процессах 0 и N дано по одному вещественному числу A. Определить для всех процессов декартову топологию в виде матрицы размера 2 × N, после чего, используя функцию MPI_Cart_sub, расщепить матрицу процессов на две одномерные строки (при этом процессы 0 и N будут главными процессами в полученных строках). Используя одну коллективную операцию пересылки данных, переслать число A из главного процесса каждой строки во все процессы этой же строки и вывести полученное число в каждом процессе (включая процессы 0 и N).
2. Краткое описание выбранного алгоритма решения задачи и листинг программы.
3. Нарисовать сеть Петри для MPI программы.
4. Распечатка результатов и времени работы программы на разном количестве процессов.
5. Построить графики зависимости времени выполнения программы от числа запущенных процессов.
6. Построить графики ускорения/замедления программы.

**Выполнение работы.**

1-2. С использованием языка программирования C++ написана параллельная программа MPI с использованием виртуальной топологии. Для работы с MPI была включена библиотека mpi.h. Для инициализации среды MPI

используется MPI_Init. MPI_Comm_size позволяет получить общее количество процессов, а MPI_Comm_rank – ранг текущего процесса. Всего процессов K, или ProcNum. Поскольку K = 2N, то переменная N принимает значение половины ProcNum. Если ранг процесса равен 0 или N, то задано по одному вещественному числу в переменной A. С помощью функции MPI_Cart_create(MPI_COMM_WORLD, 2, dims, periods, reorder, &matr_Comm) для всех процессов исходного коммуникатора MPI_COMM_WORLD задана декартова топология в виде матрицы размера 2 × N, которая определена массивом dims, определяющий размеры решетки размерностью 2. periods = {0, 0}, что означает отсутствие периодичности вдоль каждого измерения, а reorder = 0 позволяет отключить переупорядочивание процессов, matr_Comm – новый коммуникатор с декартовой топологией процессов. С помощью функции MPI_Cart_sub(matr_Comm, subdims, &row_Comm) матрица процессов из коммуникатора matr_Comm расщеплена с логическим массивом subdims = {0, 1}, показывающим входящие размерности, на две одномерные строки, у которых задается новый коммуникатор row_Comm, а 0 и N - главные процессы в полученных строках, то есть строки: процессы с рангом от 0 до N - 1 и процессы с рангом от N до ProcNum. Главные процессы, то есть с рангом 0 в новых подгруппах, с помощью коллективной операции MPI_Bcast(&A, 1, MPI_DOUBLE, 0, row_Comm) отправляют одно значение A типа MPI_DOUBLE всем процессам в соотвествующей им строке, что контролируется коммуникаторами row_Comm в каждой строке. После этого каждый процесс помечает коммуникаторы matr_Comm и row_Comm для удаления. Программа представлена в листинге 1. Пример работы программы представлен в листинге 2.

Листинг 1.

```
#include <iostream>
#include <mpi.h>
```

```
using namespace std;

int main(int argc, char** argv) {
    int ProcNum, ProcRank;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
    MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);

    int N = ProcNum / 2;
    double A;

    if (ProcRank == 0) {
        A = 1.11;
    } else if (ProcRank == N) {
        A = 5.55;
    }

    int dims[2] = {2, N};
    int periods[2] = {0, 0};
    int reorder = 0;
    MPI_Comm matr_Comm;

    double start, end;
    if (ProcRank == 0) start = MPI_Wtime();
    MPI_Cart_create(MPI_COMM_WORLD, 2, dims, periods, reorder, &matr_Comm);

    int subdims[2] = {0, 1};
    MPI_Comm row_Comm;
    MPI_Cart_sub(matr_Comm, subdims, &row_Comm);

    // int RowProcRank;
    // MPI_Comm_rank(row_Comm, &RowProcRank);
    // if (RowProcRank == 0) {
    //      cout << "Process " << ProcRank << " is the main process in its row"
<< endl;
    // }

    MPI_Bcast(&A, 1, MPI_DOUBLE, 0, row_Comm);

    MPI_Comm_free(&row_Comm);
    MPI_Comm_free(&matr_Comm);
    if (ProcRank == 0) {
            end = MPI_Wtime();
            cout << "Time " << end - start << endl;
    }

    // cout << "Process " << ProcRank << " has A = " << A << endl;

    MPI_Finalize();
    return 0;
}
```

Листинг 2.

```
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 4 --oversubscribe ./main
Process 0 is the main process in its row
Process 0 has A = 1.11
```

```
Process 1 has A = 1.11
Process 2 is the main process in its row
Process 2 has A = 5.55
Process 3 has A = 5.55
```

3. Построена сеть Петри. Сеть Петри представлена на рис. 1 и отражает параллельную программу для четырех процессов, в которой есть виртуальная топология.

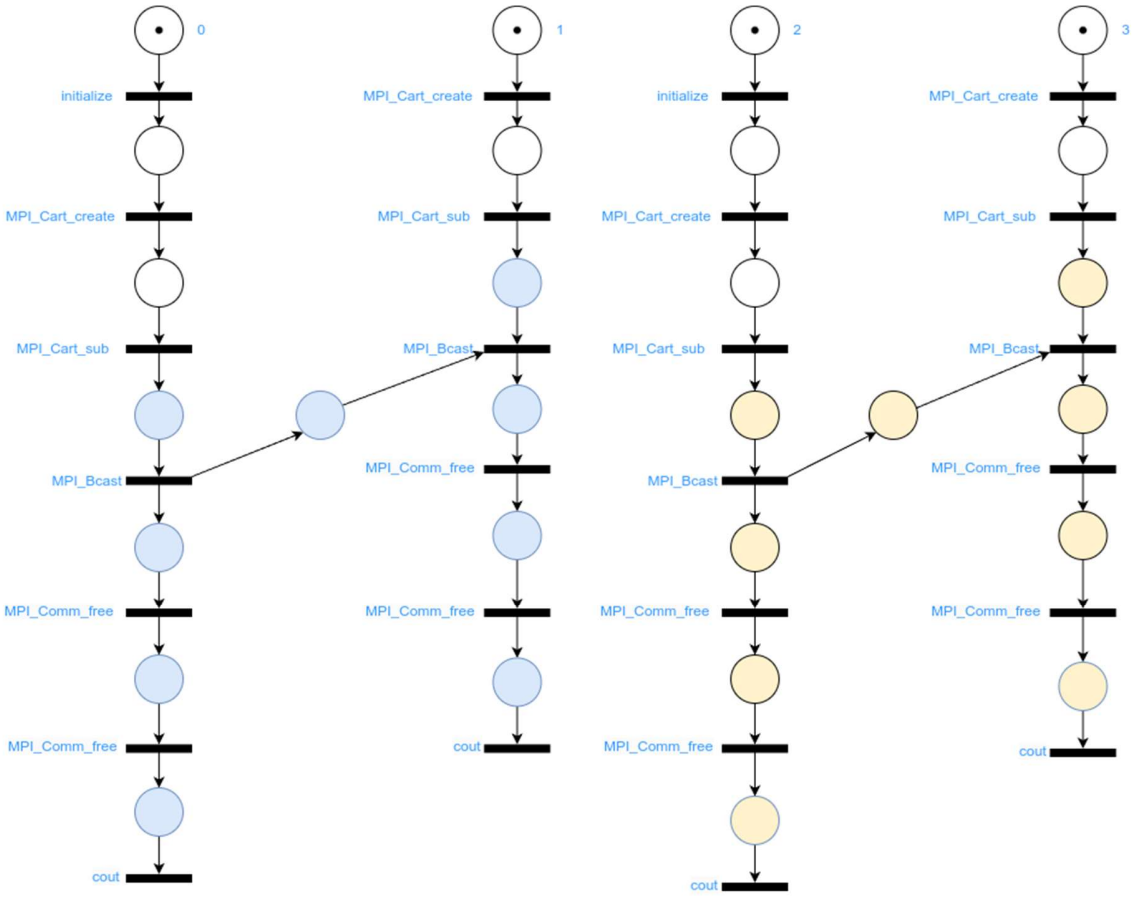


Рисунок 1 – Сеть Петри для параллельной программы с виртуальной топологией для четырех процессов

4. Программа была запущена на разных числах процессов: 2, 4, 6, 8, 10, 12, 14, 16. Время выполнения программы в зависимости от количества процессов представлено в листинге 3.

Листинг 3.

```
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 2 --oversubscribe ./main
Time 0.00034918
```

```
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 4 --oversubscribe ./main
Time 0.000501911
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 6 --oversubscribe ./main
Time 0.000563159
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 8 --oversubscribe ./main
Time 0.000709738
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 10 --oversubscribe ./main
Time 0.00128692
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 12 --oversubscribe ./main
Time 0.00153107
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 14 --oversubscribe ./main
Time 0.00198544
cut@cute:~/Документы/cpluscplus/6$ mpiexec.openmpi -n 16 --oversubscribe ./main
Time 0.00240362
```

5. На основе полученных данных при выполнении программы на разных количествах процессов построен график зависимости времени выполнения программы от числа запущенных процессов, который представлен на рис. 2 соответственно.

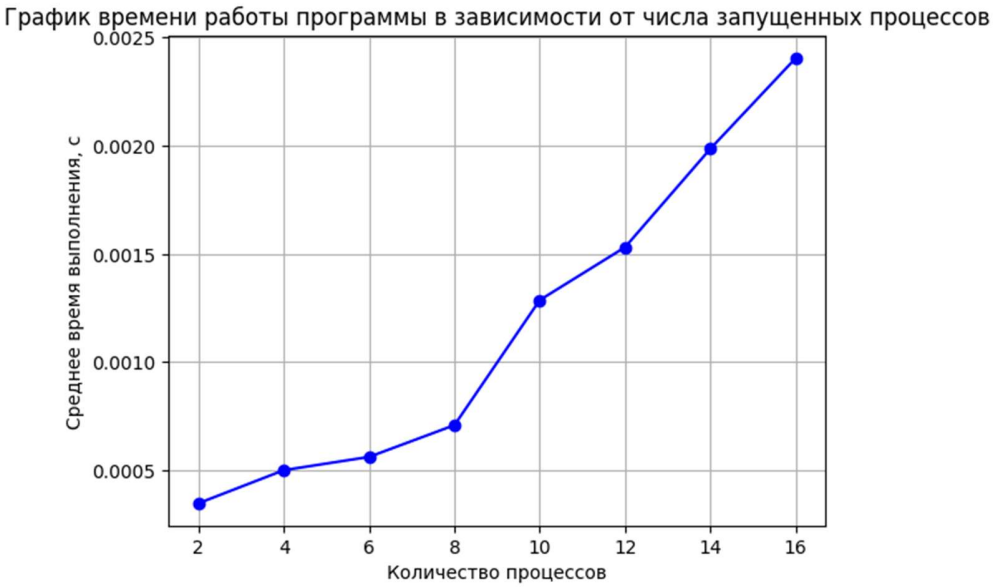


Рисунок 2 – График времени работы программы в зависимости от числа запущенных процессов с учетом суммирующей подгруппы

Исходя из графика, представленного на рис. 2, время работы программы увеличивается с ростом количества запущенных процессов. С увеличением количества процессов растет нагрузка на создание декартовой топологии, а затем на ее расщепление на две одномерные строки. Вместе с этим количество

главных процессов не меняется, однако при увеличении количества процессов увеличивается количество процессов в коллективной операции передачи данных, что нагружает программу. На результаты также влияет количество ядер и потоков в вычислительной машине. Время выполнения программы может быть разным при одинаковых условиях, что может быть вызвано затратами на создание декартовой топологии и расщеплением на подстроки, вместе с этим созданием новых коммуникаторов, передачей и приемом данных, пометками удаления новых коммуникаторов, а также разным состоянием вычислительной машины.

6. Для построения графика ускорения/замедления работы программы с разным количеством запущенных процессов необходимо разделить время выполнения программы с 2 процессами на время выполнения программы с 4, 6, 8, 10, 12, 14, 16 процессами соотвественно. График замедления представлен на рис. 3.

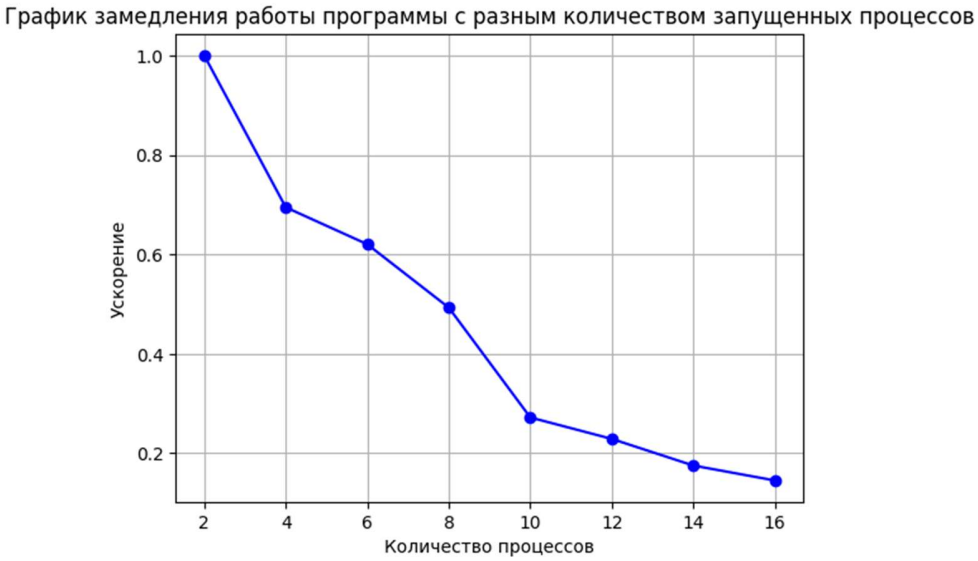


Рисунок 3 – График замедления работы программы с разным количеством запущенных процессов

**Вывод.**

Таким образом, было освоено написание параллельных программ MPI с виртуальной топологией библиотеки MPI.

Были изучены зависимости времени работы программы от числа запущенных процессов. С увеличением числа процессов время работы программы увеличивается. Увеличение времени работы программы вызвано увеличением нагрузки на создание декартовой топологии и впоследствии двух подстрок, в которых также увеличивает нагрузку коллективная операция передачи данных, и освобождением коммуникаторов. На основании полученных данных о времени работы программы построен график времени работы программы в зависимости от числа запущенных процессов и соответствующий ему график ускорения/замедления работы программы. Для программы была построена сеть Петри.