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

ОТЧЕТ

по практической работе №6
по дисциплине «Системы параллельной обработки данных»
Тема: Виртуальные топологии.

Студент гр. 0303 _____ Болкунов В.О.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург
2024

Цель работы.

Исследование и разработка параллельной программы, использующей функции построения виртуальной топологии процессов и обмен сообщениями внутри построенной группы.

Постановка задачи.

Вариант 1.

В главном процессе дано целое число N (> 1), причем известно, что количество процессов K делится на N. Переслать число N во все процессы, после чего, используя функцию **MPI_Cart_create**, определить для всех процессов декартову топологию в виде двумерной решетки — матрицы размера N × K/N (порядок нумерации процессов оставить прежним). Используя функцию **MPI_Cart_coords**, вывести для каждого процесса его координаты в созданной топологии.

Выполнение работы.

Разработана параллельно работающая программа с использованием библиотеки MPI, работающая по следующему алгоритму:

1. Каждый процесс определяет числа N и K, соответственно высоту матрицы и количество процессов.
2. Если Количество процессов не делится на высоту матрицы N, программа завершается с ошибкой.
3. С помощью метода **Cart_create** глобального коммуникатора создаётся группа процессов с топологией в виде двухмерной решётки.
4. Каждый процесс выводит свой ранг и координаты внутри своей топологии.

Сеть петри с 3 процессами для данного алгоритма представлена на рисунке 1.

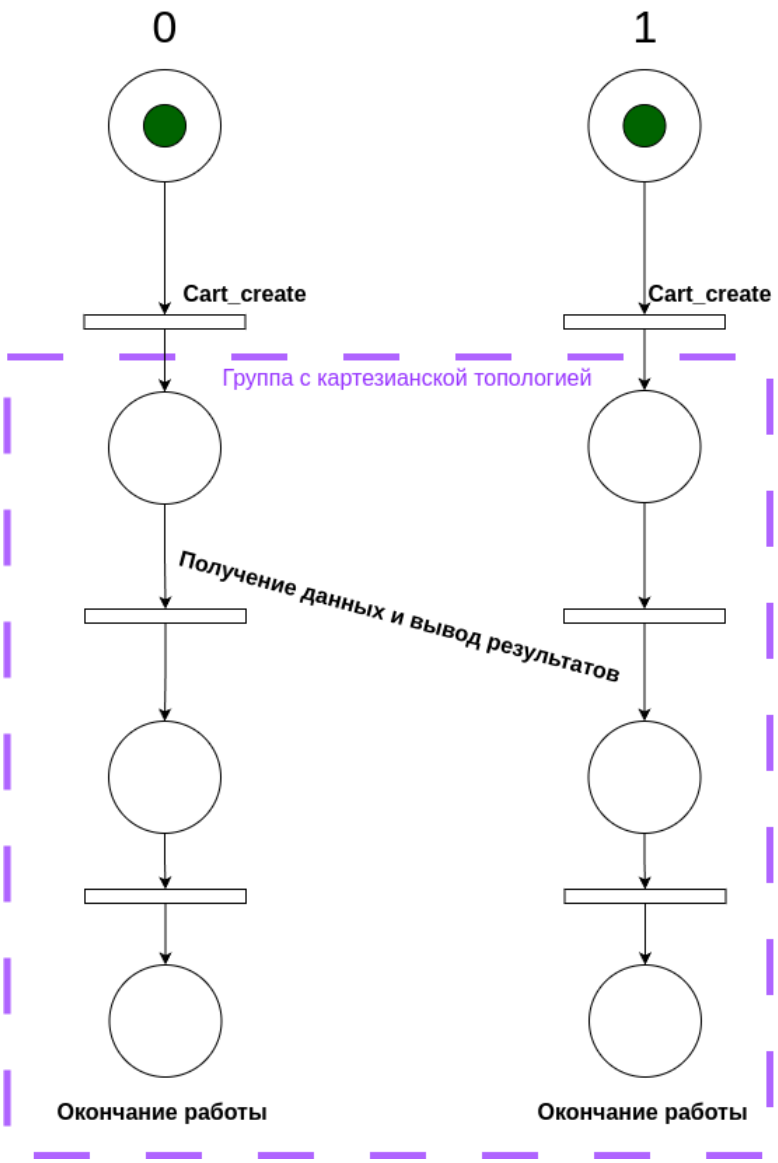

Рисунок 1. Сеть петри

Пример работы программы с 9 процессами представлен на рисунке 2. Исходный код программы представлен в приложении А.

```
[5] pos = {1, 2}; time = 0.00132135
vlad@vlad-pc:~/projects/SPOD/lab6$ ./run.sh 9 3
[6] pos = {2, 0}; time = 0.000368886
[7] pos = {2, 1}; time = 0.000400585
[8] pos = {2, 2}; time = 0.000330683
[0] pos = {0, 0}; time = 0.000468693
[1] pos = {0, 1}; time = 0.000401426
[2] pos = {0, 2}; time = 0.00044565
[3] pos = {1, 0}; time = 0.000464105
[4] pos = {1, 1}; time = 0.000459155
[5] pos = {1, 2}; time = 0.000393231
vlad@vlad-pc:~/projects/SPOD/lab6$
```

Рисунок 2. Пример работы программы

В данном примере для для каждого процесса выводится ранг в группе и позиция внутри топологии.

Программа была запущена на разном количестве процессоров: от 3 до 30 с шагом 3 и высотой матрицы 3. Время работы программы в данной задаче соответствует максимальному времени работы из всех процессов.

На рисунке 3 представлен график зависимости времени работы программы от количества задействованных процессоров. На рисунке 4 представлен график ускорения.

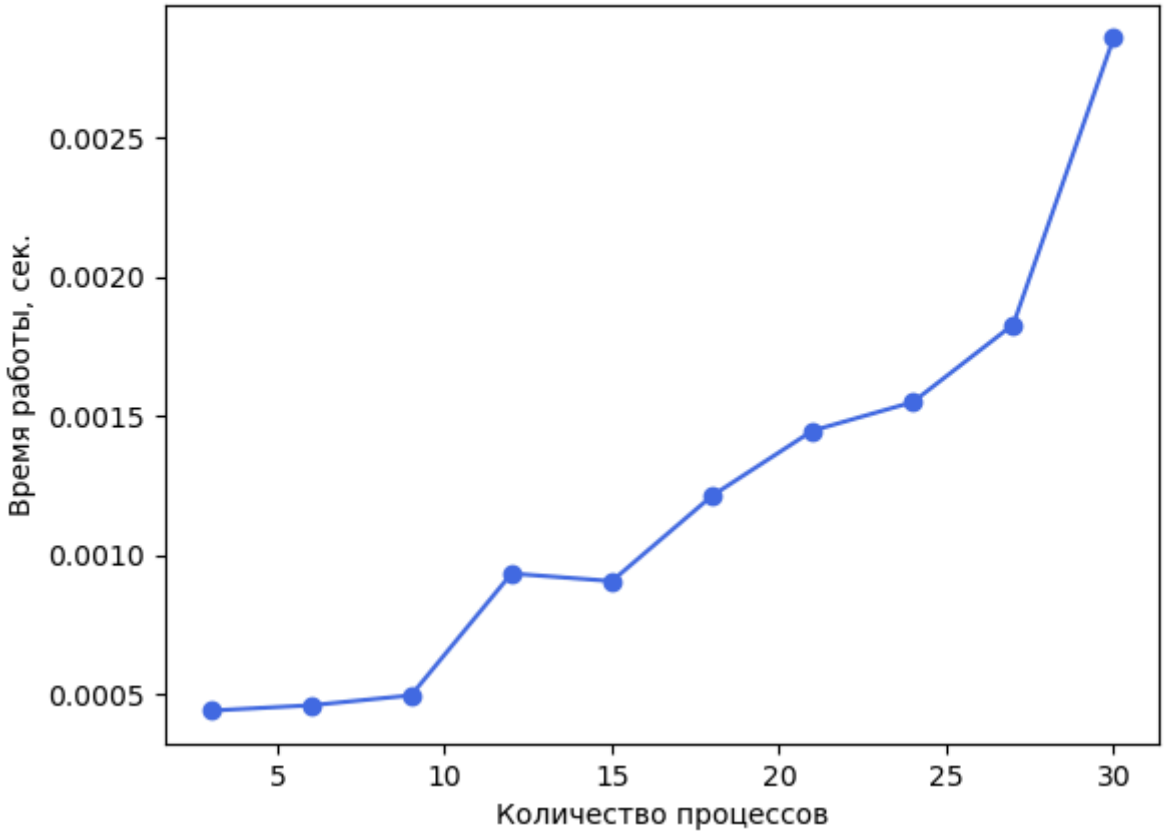

Рисунок 3. График времени работы программы

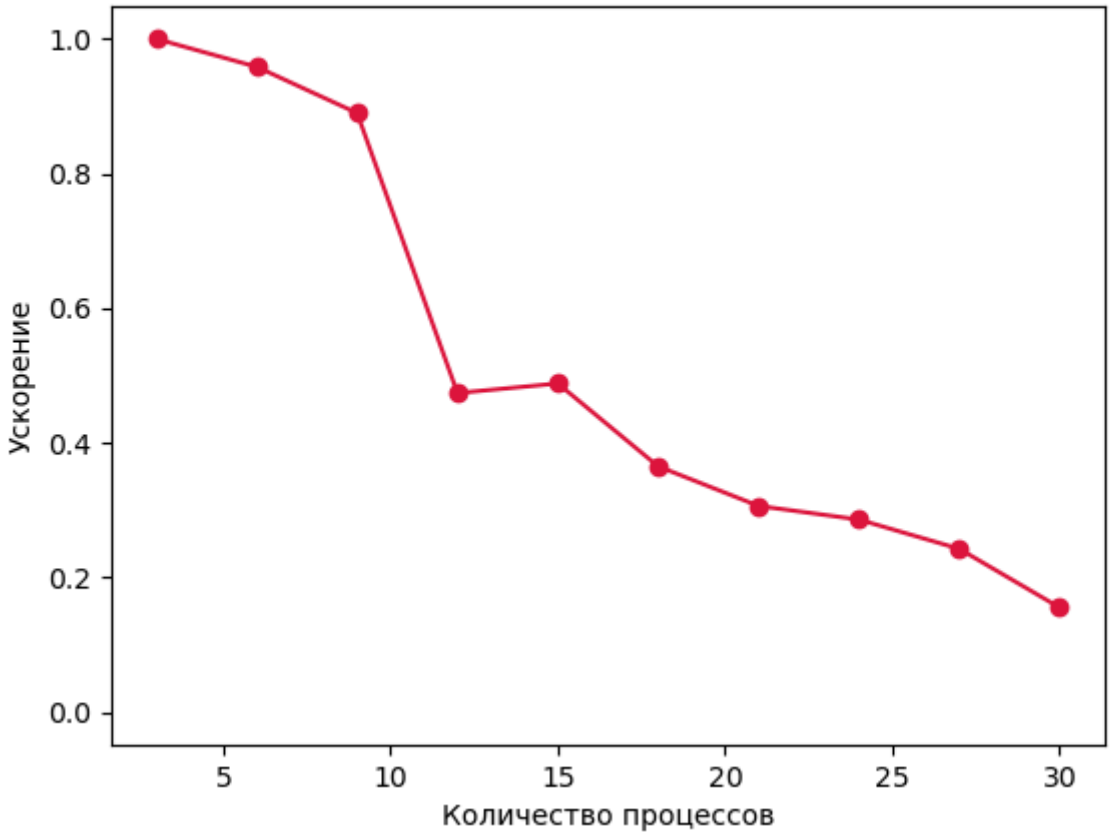

Рисунок 4. График ускорения программы

По данным графикам видно, что добавление дополнительных процессов замедляет работу программы, так в данной задаче отсутствует параллельно выполняемый код и большая часть времени работы программы уходит на формирование группы процессов и виртуальной топологии.

Выводы.

В результате выполнения работы была разработана параллельная программа, в которой создаётся новая группа процессов на основе глобальной группы с виртуальной топологией в виде двумерной решётки, и выводятся координаты процессов в построенной топологии. Разработанная программа запущена на разном количестве процессоров: от 3 до 30. В результате тестирования программы построены графики времени работы программы и ускорения.

Увеличение количество процессов привело к замедлению работы программы, так как в решаемой задаче большую часть времени работы программы занимает время формирования группы процессов и виртуальной топологии, и при этом отсутствуют параллельно выполняемые задачи.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

Файл main.cpp

```cpp
#include <mpi.h>
#include <array>
#include <ctime>
#include <iostream>
#include <vector>

int main(int argc, char **argv) {
 MPI::Init(argc, argv);
 MPI::Intracomm &worldComm = MPI::COMM_WORLD;

 const int N = std::atoi(argv[1]);
 const int K = worldComm.Get_size();

 if (K % N != 0) {
   throw MPI::Exception(MPI::ERR_ARG);
 }

 double time = 0, start, end;

 int W = N;
 int H = K / N;

 const int dims[] = {W, H};
 const bool periods[] = {false, false};
 int coords[2], rank;

 start = MPI::Wtime();

 MPI::Cartcomm cartComm = worldComm.Create_cart(2, dims, periods, false);
 rank = cartComm.Get_rank();
 cartComm.Get_coords(rank, 2, coords);

 end = MPI::Wtime();
 time += end - start;

 std::cout << time << ',' << std::endl;

 MPI::Finalize();
}
```

Файл build.sh

```
mpicxx.openmpi main.cpp -o lab6
```

Файл run.sh

```
mpiexec.openmpi -n $1 --oversubscribe ./lab6 $2
```