**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Параллельные алгоритмы»**

ТЕМА: **Виртуальные топологии.**

|  |  |  |
| --- | --- | --- |
| Студент |  | Степаненко Д. В. |
| Преподаватель |  | Татаринов Ю. С. |

Санкт-Петербург

2023 г.

## Цель

Исследовать виртуальные топологии в библиотеки MPI, с использованием функцией для ее создания, а также рассмотреть операции с топологиями, в частности сдвиг.

## Постановка задачи (вариант 10)

Число процессов К является четным: K = 2N, N > 1; в каждом процессе дано вещественное число A. Определить для всех процессов декартову топологию в виде матрицы размера 2 × N (порядок нумерации процессов оставить прежним) и для каждой строки матрицы осуществить циклический сдвиг исходных данных с шагом 1 (число A из каждого процесса, кроме последнего в строке, пересылается в следующий процесс этой же строки, а из последнего процесса — в главный процесс этой строки). Для определения рангов посылающих и принимающих процессов использовать функцию MPI\_Cart\_shift, пересылку выполнять с помощью функции MPI\_Sendrecv. Во всех процессах вывести полученные данные.

## Выполнение работы

Первоначально происходит инициализация MPI, определяются общее количество процессов и ранг текущего процесса. Затем определяются значение переменной N - количество строк в декартовой топологии (количество всех процессов/2). Переменная A принимает значение ранга текущего процесса.

Далее происходит создание декартовой топологии с помощью функции MPI\_Cart\_create(). После определяются переменные recv\_data, source и destination. Переменная recv\_data будет хранить принятые данные от другого процесса при сдвиге, а переменные source и destination - ранг исходного и целевого процессов при передаче данных.

Далее вызывается функция MPI\_Cart\_shift(), которая определяет источник и цель передачи данных в декартовой топологии. Вызывается функция MPI\_Sendrecv(), которая осуществляет пересылку своего ранга вдоль оси y к соответствующему процессу и получает ранг от другого процесса. Под конец освобождается коммуникатор, выводится информация о полученных данных с помощью функции, и освобождаются ресурсы системы.

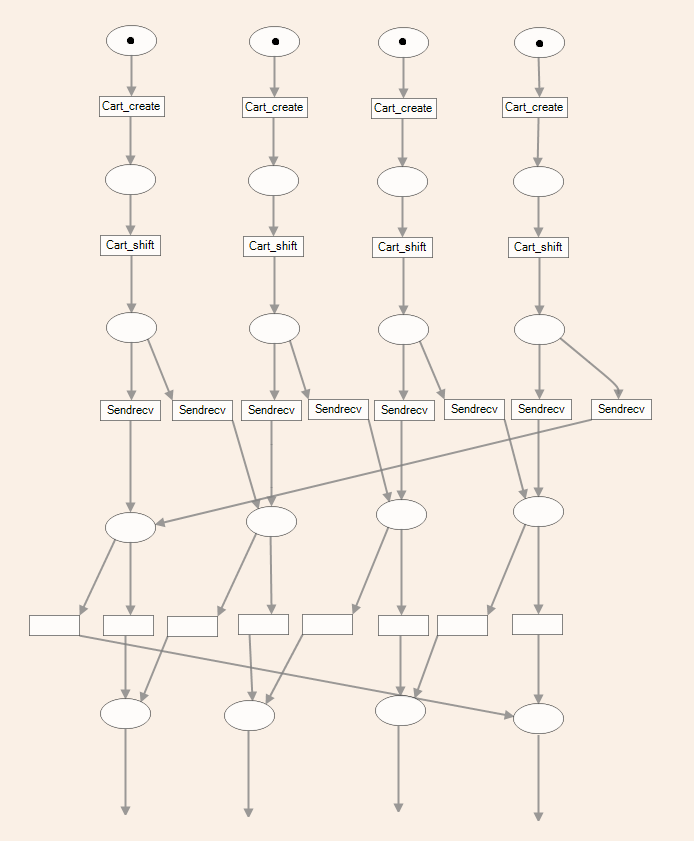


Рисунок 1 - Сеть Петри основной параллельной части программы для четырех процессов.

Листинг программы:

#include <stdio.h>

#include <mpi.h>

int main(int argc, char\*\* argv) {

int size, rank;

MPI\_Init(&argc, &argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

int N = size / 2;

int A = rank;

// Создание декартовой топологии

int dims[2] = {2, N};

int periods[2] = {0, 1}; //для передачи по строке от последнего к главному

int reorder = 0;

MPI\_Comm cart\_comm;

// Аргументы: коммуникатор; размерность; массив размеров сетки; массив периодов; переупорядочивание; новый коммуникатор;

MPI\_Cart\_create(MPI\_COMM\_WORLD, 2, dims, periods, reorder, &cart\_comm);

int recv\_data;

int source;

int destination;

// Аргументы: коммуникатор; направление 0 - вверх, 1 - по у; размер смещения; источник; назначение;

MPI\_Cart\_shift(cart\_comm, 1, 1, &source, &destination);

MPI\_Sendrecv(&A, 1, MPI\_INT, destination, 0, &recv\_data, 1, MPI\_INT, source, 0, cart\_comm, MPI\_STATUS\_IGNORE);

MPI\_Comm\_free(&cart\_comm);

printf("Процесс %d: было A = %d, получил A = %d\n", rank, A, recv\_data);

MPI\_Finalize();

return 0;

}

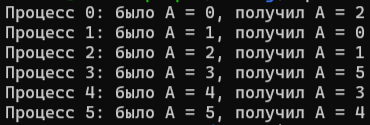


Рисунок 2 - Полученный вывод при запуске программы на шести процессах.

Таблица 1 – Результаты работы программы на разном количестве процессов.

| Количество процессов (шт) | Среднее затрачиваемое время (мс) |
| --- | --- |
| 2 | 0,0508 |
| 4 | 0,0782 |
| 6 | 0,0939 |
| 8 | 0,1416 |
| 10 | 0,1556 |
| 12 | 0,1963 |

Расчеты ускорения программы выполним по формуле:

Таблица 2 – Результаты расчетов ускорения программы.

| Количество процессов P (шт) | Ускорение |
| --- | --- |
| 2 | 1 |
| 4 | 0,65 |
| 6 | 0,541 |
| 8 | 0,359 |
| 10 | 0,326 |
| 12 | 0,259 |

## Вывод

В ходе выполнения лабораторной работы были изучены операции с созданием и работой с топологиями в библиотеке MPI, использованы на практике функции *MPI\_Cart\_create() и MPI\_Cart\_shift()*. Первая – создает новую топологию по заданным параметрам (размерность, периодичность, перемешивание) на основе нового коммуникатора. Вторая – возвращает координаты, которые можно будет использовать для циклического сдвига. Тоже имеет настраиваемые параметры: направление и размер смещения. Для пересылки сообщений по заданным координатам использовалась блокируемая функция *MPI\_Sendrecv().* Таким образом, была написана программа, удовлетворяющая ТЗ.

Время выполнения программы засекалось на параллельной части кода, где использовались вышеперечисленные функции. Количество процессов изменялось от 1 до 12, т.к. дальнейшее увеличение было бы малоинформативным. На рисунке 1 видно, что скорость выполнения программы увеличивается линейно. Это происходит потому, что с добавлением новой пары процессов длина строки (по оси y) увеличивается на 1. Также количество процессов N влияет на скорость создания новой топологии. Следовательно, скорость исполнения программного кода будет пропорционально количеству процессов.

Исходя из времени выполнения программы, можно сделать вывод, что и ускорение будет уменьшаться с увеличением процессов. Эту зависимость можно посмотреть на графике.