**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).

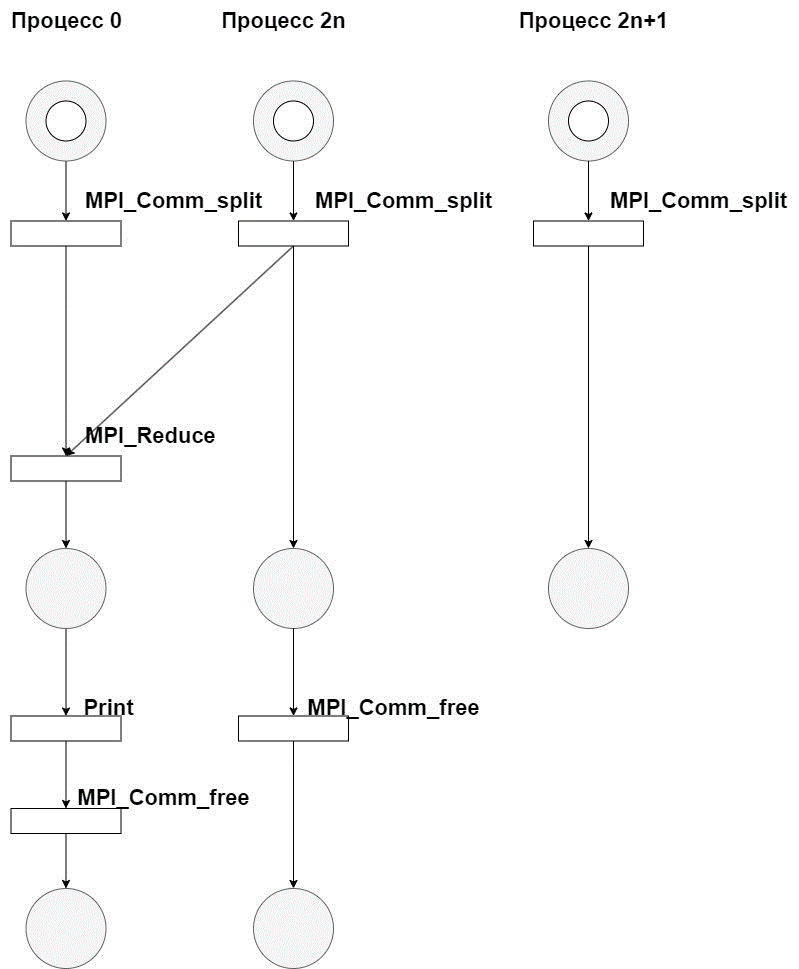


Рисунок 1 — Сеть Петри основной части алгоритма

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.с**

**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:**

Sp (𝑛) = 𝑇1(𝑛)/𝑇p(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.