**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

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

**Тема: «Коллективные операции»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9303 |  | Королёв С.Ю. |
| Преподаватель |  | Татаринов Ю.С. |

Санкт-Петербург

2023

**Цель работы**

Базовое освоение и изучение коллективных операций в библиотеке MPI.

**Формулировка задания**

В главном процессе дан набор из 3K чисел, где K — количество процессов. Используя функцию MPI\_Scatter, переслать по 3 числа в каждый процесс (включая главный) и вывести в каждом процессе полученные числа.

**Ход работы**

Для изучения коллективных операций в библиотеке MPI был создан скрипт на языке программирования С++ для запуска и обработки результатов запусков процессов.

**Листинг итоговой программы**

#include <iostream>

#include "mpi.h"

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

MPI\_Init(&argc, &argv);

int rank, size;

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

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

int data\_size = 3 \* size;

int \*data\_array = new int[data\_size];

int recvBuff[3];

for (int i = 0; i < data\_size; i++) {

data\_array[i] = rand() % 100;

}

MPI\_Scatter(data\_array, 3, MPI\_INT, recvBuff, 3, MPI\_INT, 0, MPI\_COMM\_WORLD);

printf("Array by process %d: ", rank);

for (int i = 0; i < 3; i++) {

printf("%d ", recvBuff[i]);

}

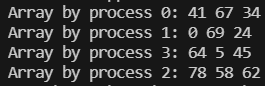
MPI\_Finalize();

return 0;

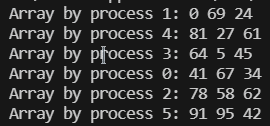
}

**Результаты работы исходной программы на 4, 6, 8, 10 процессах**

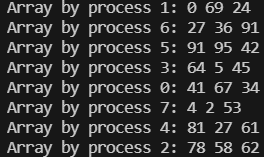
На 4 процессах:



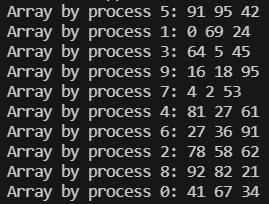
На 6 процессах:



На 8 процессах:



На 10 процессах:

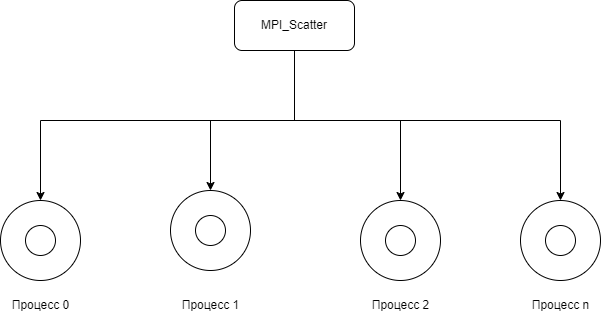


**Краткое описание выбранного алгоритма решения**

Распределяет данные от одного члена по всем членам группы, т.е. производит обобщённую передачу. MPI\_Scatter выполняет обратную операцию, выполняемую функцией MPI\_Gather.

Массив data\_array размерности data\_size (3 \* число процессов) через цикл for заполняется случайно сгенерированными числами в диапазоне от 0 до 99. Далее функция MPI\_Scatter распределяет данные из массива data\_array по буферам recvBuff[3] каждому процессу, по 3 элемента массива data\_array для каждого процесса соответственно.

**Формальное описание выбранного алгоритма решения с использованием аппарата сетей Петри**



**График зависимости времени выполнения программы от числа процессов для разного объема исходных данных**

Таблица 1. Зависимость времени выполнения программы от количества процессов.

|  |  |
| --- | --- |
| Число процессов | Время выполнения для итоговой программы (мс.) |
| 4 | 1.126 |
| 6 | 1.837 |
| 8 | 3.129 |
| 10 | 4.762 |

**Вывод**

В ходе выполнения работы были получены навыки использования коллективных операций в библиотеке MPI.

Был построен график зависимости времени выполнения программы от количества процессов. По графику можно сделать вывод, что время выполнения для итоговой программы прямо пропорционально числу процессов. Это объясняется тем, что с увеличением числа процессов увеличивается и затрачиваемое время для отправки элементов массива data\_array, потому что количество элементов в этом массиве увеличивается.