**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №1**

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

**Тема: «Использование функций обмена данными «точка-точка» в библиотеке MPI»**

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

Санкт-Петербург

2023

**Цель работы**

Базовое освоение и изучение функций обмена данными «точка-точка» в библиотеке MPI.

**Формулировка задания**

Сдвиг массива, распределенного между узлами. Процесс 0 генерирует массив и раздает его другим процессам, после чего выполняется циклический сдвиг массива.

**Ход работы**

Для изучения использования функций обмена данными «точка-точка» в библиотеке MPI в процессе работы был создан скрипт на языке программирования C++ для запуска и обработки результатов запусков процессов.

**Листинг итоговой программы:**

#include <iostream>

#include <mpi.h>

using namespace std;

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

MPI\_Init(&argc, &argv);

int rank, size;

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

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

int originalArray[] = { 1, 2, 3, 4, 5 };

const int arraySize = sizeof(originalArray) / sizeof(int);

int shiftedArray[arraySize];

MPI\_Status status;

double start = MPI\_Wtime();

if (size < 2) {

std::cout << "This program requires at least 2 processes to run." << std::endl;

MPI\_Finalize();

return 1;

}

if (rank == 0) {

cout << "Original array: [";

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

cout << originalArray[i] << " ";

}

cout << "]" << endl;

// Send the original array to process 1

MPI\_Send(originalArray, arraySize, MPI\_INT, 1, 0, MPI\_COMM\_WORLD);

// Receive the shifted array from the last process

MPI\_Recv(shiftedArray, arraySize, MPI\_INT, size - 1, 0, MPI\_COMM\_WORLD, &status);

cout << "Resulted array: [";

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

cout << shiftedArray[i] << " ";

}

cout << "]" << endl;

}

else {

// Receive the array from the previous process

MPI\_Recv(shiftedArray, arraySize, MPI\_INT, rank - 1, 0, MPI\_COMM\_WORLD, &status);

cout << "Process " << rank << " received array: [";

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

cout << shiftedArray[i] << " ";

}

cout << "]" << endl;

// Shift the array to the left

int temp = shiftedArray[0];

for (int i = 0; i < arraySize - 1; i++) {

shiftedArray[i] = shiftedArray[i + 1];

}

shiftedArray[arraySize - 1] = temp;

if (rank == size - 1) {

// Send the shifted array to process 0

MPI\_Send(shiftedArray, arraySize, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

else {

// Send the shifted array to the next process

MPI\_Send(shiftedArray, arraySize, MPI\_INT, rank + 1, 0, MPI\_COMM\_WORLD);

}

}

double deltaTime = MPI\_Wtime() - start;

if (rank == 0) {

cout << "Time delta: " << deltaTime;

}

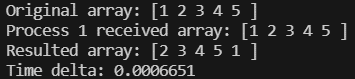
MPI\_Finalize();

return 0;

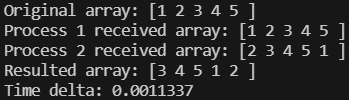
}

**Результаты работы итоговой программы на 2, 3, 4, 5, 6 процессах:**

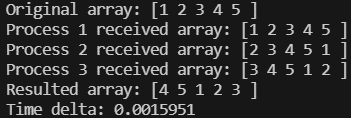
На 2 процессах:



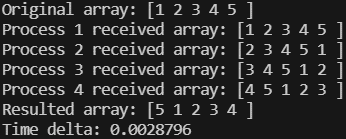
На 3 процессах:



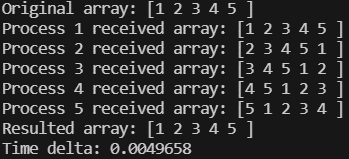
На 4 процессах:



На 5 процессах:



На 6 процессах:



Стоит заметить, что не имеет смысла выполнять операцию циклического сдвига массива в том случае, если количество процессов менее 2 (то есть существует только процесс с рангом 0). Это связано с тем, что процесс с рангом 0 не сможет отправлять данные другим процессам, чтобы реализовать циклический сдвиг, потому что других процессов не будет существовать.

**Краткое описание выбранного алгоритма решения**

Процесс с рангом 0 выводит оригинальный массив и отправляет его процессу с рангом 1. Процесс с рангом 1 получает от процесса с рангом 0 массив и выводит его, после чего выполняет циклический сдвиг массива влево по следующей логике:

- сохраняется значение первого элемента массива во временную переменную temp;

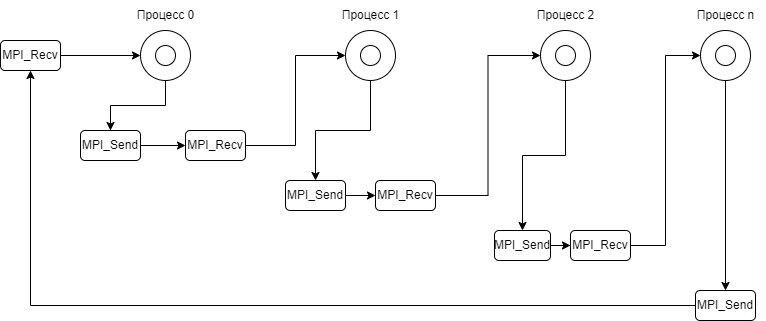
- цикл for итерируется по элементам массива, начиная с нулевого индекса и до индекса предпоследнего элемента массива; в этом цикле каждый элемент массива заменяется значением следующего элемента массива, таким образом все элементы массива смещаются влево на 1 позицию, кроме последнего элемента;

- последнему элементу массива присваивается значение из временной переменной temp, при этом первый элемент массива помещается в конец, завершая операцию циклического сдвига влево.

Далее процесс с рангом 1 отправляет получившийся массив процессу с рангом 2. Процесс с рангом 2 выводит полученный от процесса с рангом 1 массив, после выполняется та же операция циклического сдвига массива влево, и так далее по аналогии до того момента, пока мы не дойдем до последнего процесса, который уже отправляет итоговый массив процессу с рангом 0.

В конце процесс с рангом 0 выводит итоговый массив, получившихся после всех преобразований.

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

****

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

Таблица 1. Зависимость времени выполнения программы от количества процессов.

|  |  |
| --- | --- |
| Число процессов | Время выполнения для итоговой программы (мс.) |
| 2 | 0,6651 |
| 3 | 1,1337 |
| 4 | 1,5951 |
| 5 | 2,8796 |
| 6 | 4,9658 |

**Вывод**

В ходе выполнения работы были получены навыки использования функций обмена данными «точка-точка» в библиотеке MPI.

Был построен график зависимости времени выполнения 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 количество операций циклического сдвига массива, на которые затрачивается больше времени.