**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №0**

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

**Тема: «Запуск параллельной программы на различном числе одновременно работающих процессов, упорядочение вывода результатов»**

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

Санкт-Петербург

2023

**Цель работы**

Базовое освоение и изучение возможностей MPI, выявление особенностей работы.

**Формулировка задания**

Запустить программу на 1,2 … N процессах несколько раз. Проанализировать порядок вывода сообщений на экран. Вывести правило, определяющее порядок вывода сообщений. Модифицировать программу таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего процесса

**Ход работы**

Для выявления закономерностей окончания работы потоков в MPI были созданы скрипты на языке программирования C++ для запуска и обработки результатов запусков процессов.

**Листинг исходной программы**:

#include <iostream>

#include "mpi.h"

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

int ProcNum, ProcRank, RecvRank;

MPI\_Status Status;

MPI\_Init(&argc, &argv);

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

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

double start = MPI\_Wtime();

if (ProcRank == 0) {

printf("Hello from process %3d\n", ProcRank);

double timeFprocess = MPI\_Wtime() - start;

for (int i = 1; i < ProcNum; i++) {

MPI\_Recv(&RecvRank, 1, MPI\_INT, MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &Status);

printf("Hello from process %3d\n", RecvRank);

}

}

else {

MPI\_Send(&ProcRank, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

double deltaTime = MPI\_Wtime() - start;

if (ProcRank == 0) {

printf("Time delta: %f\n", deltaTime);

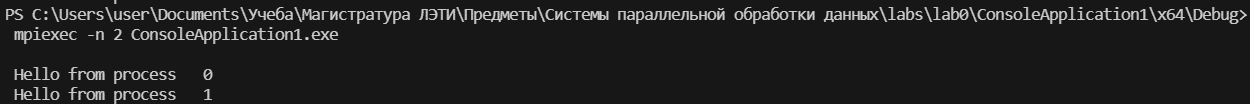
}

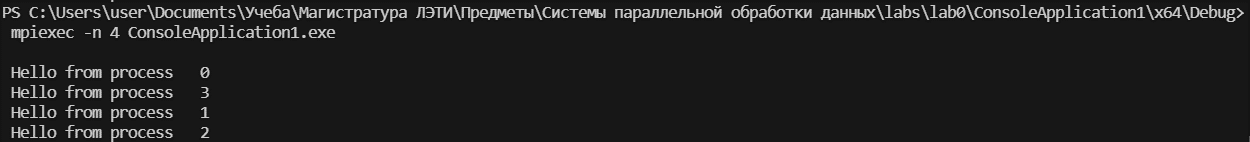
MPI\_Finalize();

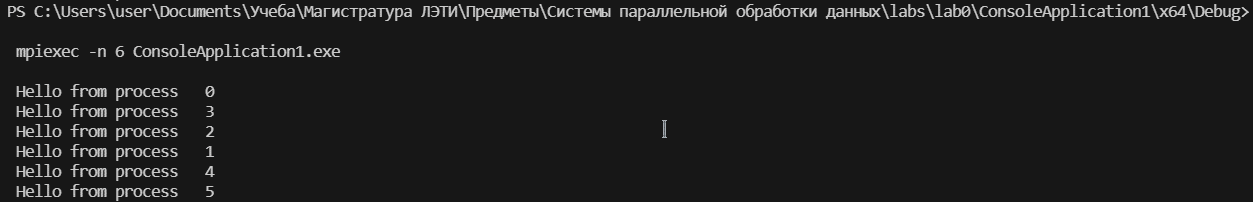
return 0;

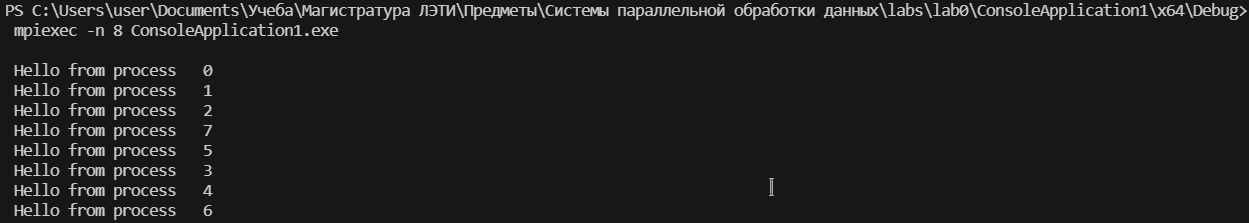
}

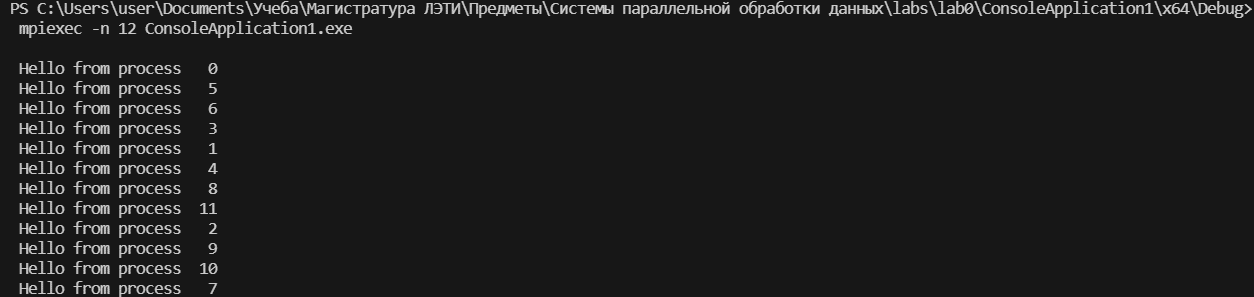
**Результаты работы исходной программы на 2, 4, 6, 8, 12 процессах:**











Можно заметить, что первым всегда выводится сообщение от потока 0, а остальные сообщения от потоков выводятся в непоследовательном порядке. Вероятным местом для вывода сообщения i является i-ое место.

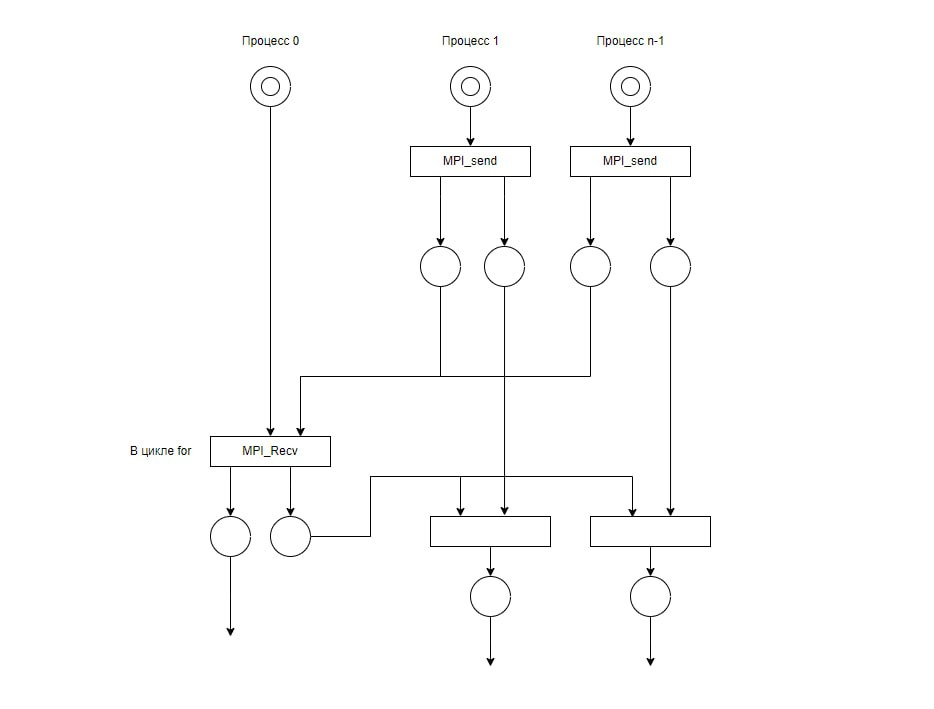
**Краткое описание выбранного алгоритма решения.**

Для того, чтоб порядок вывода сообщений соответствовал номеру потока, нужно в функции MPI\_Recv заменить аргумент source со значения

MPI\_ANY\_SOURCE на переменную i, являющуюся счетчиком цикла.

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

После внесения вышеописанного изменения нулевой процесс будет принимать сообщения по порядку от 1 до n – 1.



**Листинг измененной программы**:

#include <iostream>

#include "mpi.h"

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

int ProcNum, ProcRank, RecvRank;

MPI\_Status Status;

MPI\_Init(&argc, &argv);

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

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

double start = MPI\_Wtime();

if (ProcRank == 0) {

printf("Hello from process %3d\n", ProcRank);

double timeFprocess = MPI\_Wtime() - start;

for (int i = 1; i < ProcNum; i++) {

MPI\_Recv(&RecvRank, 1, MPI\_INT, i, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &Status);

printf("Hello from process %3d\n", RecvRank);

}

}

else {

MPI\_Send(&ProcRank, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

double deltaTime = MPI\_Wtime() - start;

if (ProcRank == 0) {

printf("Time delta: %f\n", deltaTime);

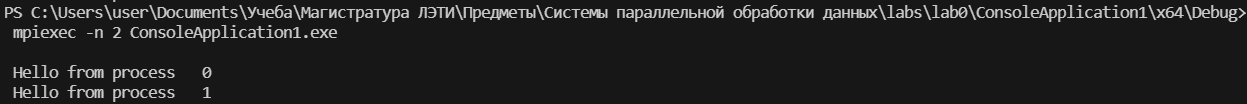
}

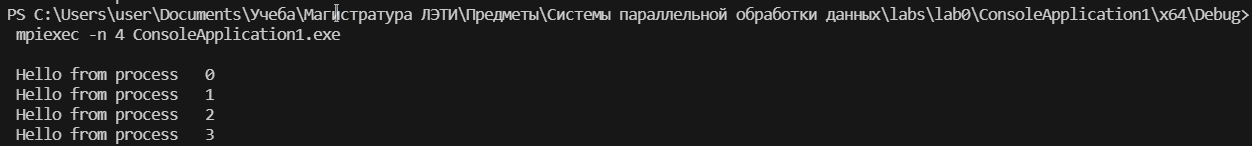
MPI\_Finalize();

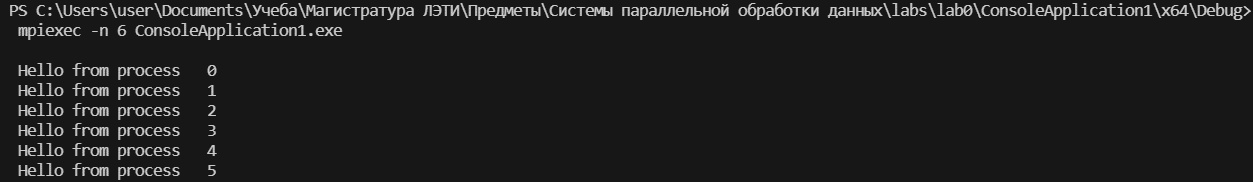
return 0;

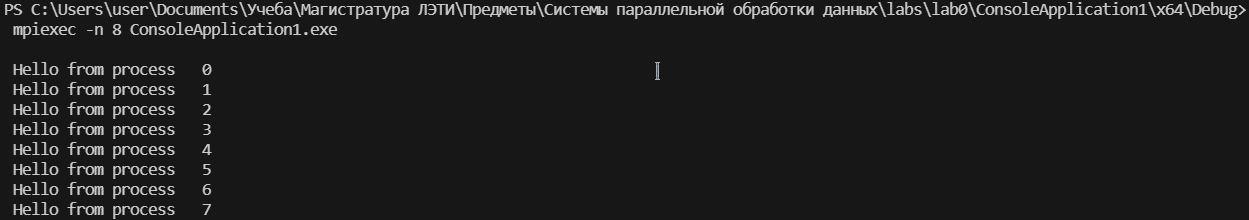
}

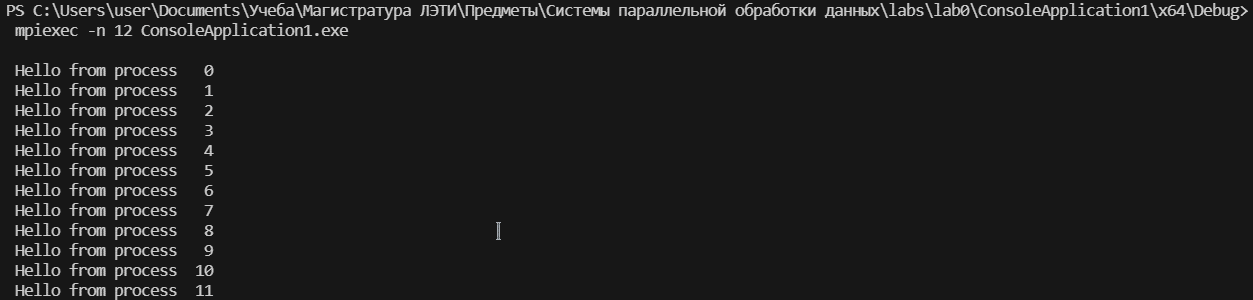
**Результаты работы измененной программы на 2, 4, 6, 8, 12 процессах:**











Таким образом, можно заметить, что теперь порядок вывода сообщений соответствует номеру потока.

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

Таблица 1. Зависимость времени выполнения от количества процессов

|  |  |  |
| --- | --- | --- |
| Число процессов | Время для исходной программы (мс.) | Время для измененной программы (мс.) |
| 2 | 0.221 | 0.341 |
| 4 | 0.537 | 0.453 |
| 6 | 0.549 | 1.401 |
| 8 | 0.693 | 0.979 |
| 12 | 1.374 | 0.675 |

**Вывод**

В ходе выполнения работы были получены навыки разработки и запуска параллельной программы с использованием технологии MPI для разного числа одновременно работающих процессов. Программа была модифицирована так, чтобы порядок вывода сообщений соответствовал номеру соответствующего процесса.

Был построен график зависимости времени выполнения от количества процессов до и после изменения программы: после внесенных изменений программа выполняется медленнее. Это происходит потому, что нулевой процесс ожидает сообщение от процесса с определенным номером, из-за этого происходит задержка.

При выполнении на 8 и 12 процессах измененная программа работает быстрее. Это связано с тем, в целях достижения последовательного вывода некоторые процессы блокируются и перестают конкурировать между собой.