МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Параллельные алгоритмы»
Тема: Запуск параллельной программы на различном числе одновременно работающих процессов, упорядочение вывода результатов.

Студент гр. 3384 _____ Рудаков А.Л.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург
2025

Цель работы.

Изучение основ MPI и использование их на практике, путем написания параллельных программ, запускаемых на различном числе одновременно работающих процессов

Задание.

Задание 1 Все процессы, кроме процесса с рангом 0, передают значение своего ранга нулевому процессу. Процесс с рангом 0 сначала печатает значение своего ранга, а далее принимает сообщения с рангами процессов и также печатает их значения. При этом важно отметить, что порядок приема сообщений заранее не определен и зависит от условий выполнения параллельной программы (более того, этот порядок может изменяться от запуска к запуску).

2. Проанализировать порядок вывода сообщений на экран. Для этого запустит программу несколько раз при фиксированном числе процессов, например -8ми. Вывести правило, определяющее порядок вывода сообщений.

3. Выполнить требования к содержанию отчета для задания 1

Задание 2: Модифицировать программу таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего процесса.

1. Выполнить требования к содержанию отчета для задания 2

2. Сравнить результаты работы двух программ.

Выполнение работы.

Для выполнения первого задания написан код на языке программирования C++, при помощи средств MPI разделяющий программу на параллельную между функциям *MPI_Init(&argc, &argv)* и *MPI_Finalize(),* а также реализующий для каждого процесса определение его ранга, посредством функции *MPI_Comm_rank(MPI_COMM_WORLD, &proc_rang)* и общее количество процессов при помощи функции

MPI_Comm_size(MPI_COMM_WORLD, &num_proc). Кроме этого реализована пересылка данных о ранге 0-ому процессу, если его собственный ранг не равен 0, с использованием функции *MPI_Send(&proc_rang, 1, MPI_INT, 0, 0, MPI_COMM_WORLD)*, а также вывод информации о запущенном процессе с рангом 0 и его прием сообщений от других процессов посредством функции *MPI_Recv(&recv_rang,1, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status)*.

Схема Петри представлена на рис.1.

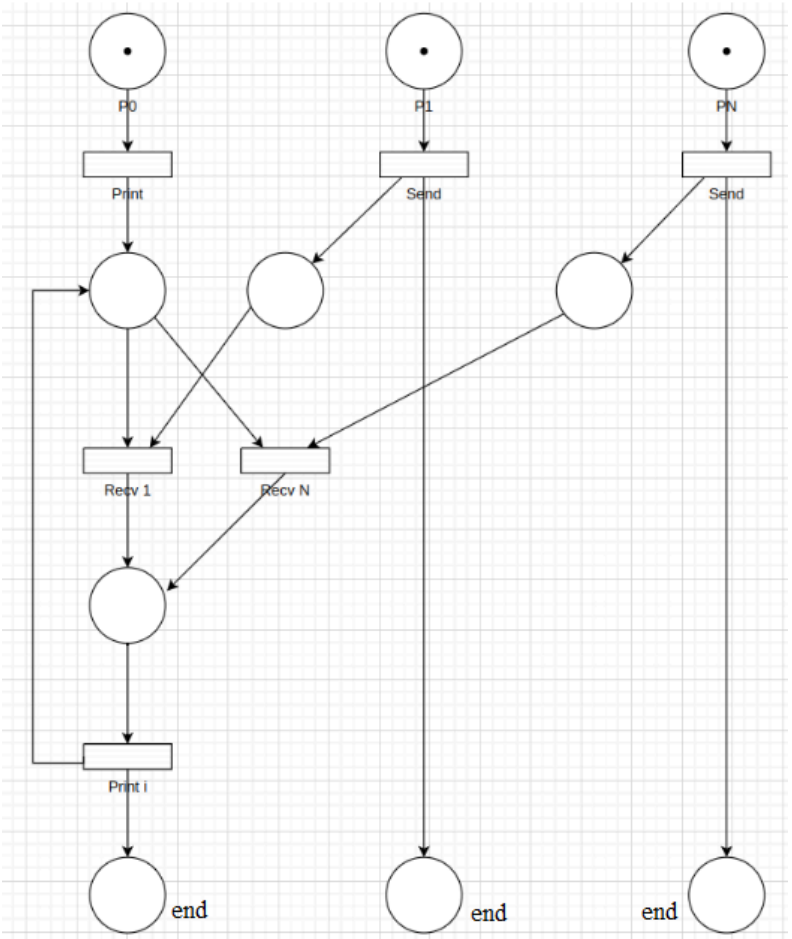

Рисунок 1 — Схема Петри.

Код, написанный для задания 1 на языке программирования C++ см. Приложение А.

Результаты работы программы см. Приложение В.

Зависимость времени работы программы для задания номер 1 от количества процессов можно увидеть в табл.1 и на рис.2.

Табл.1 см. Приложение Б.

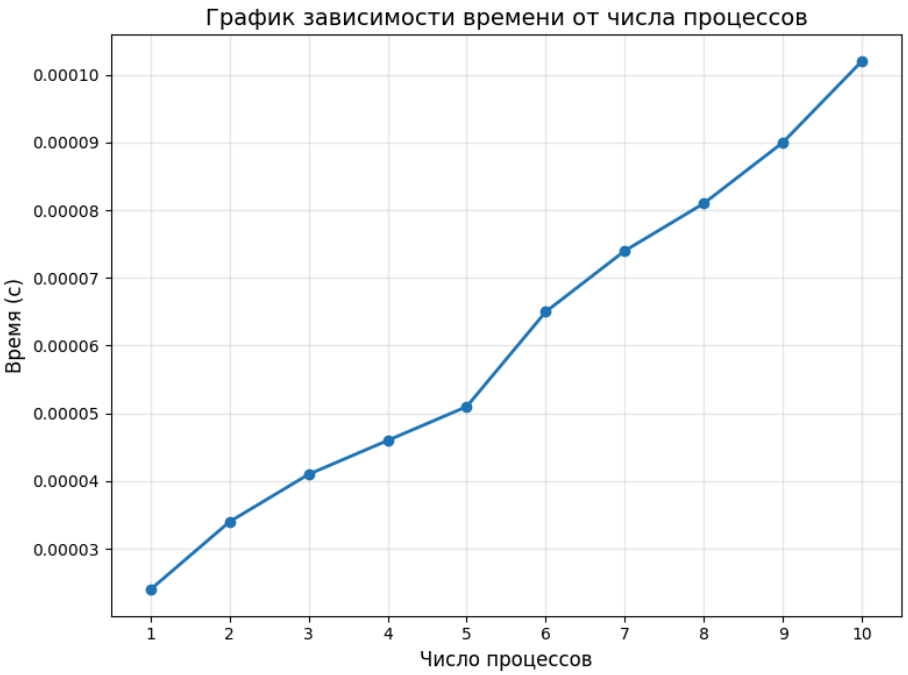

Рисунок 2 - График зависимости времени от числа процессов

На рис.3 изображен график зависимости ускорения от числа процессов для задачи 1.

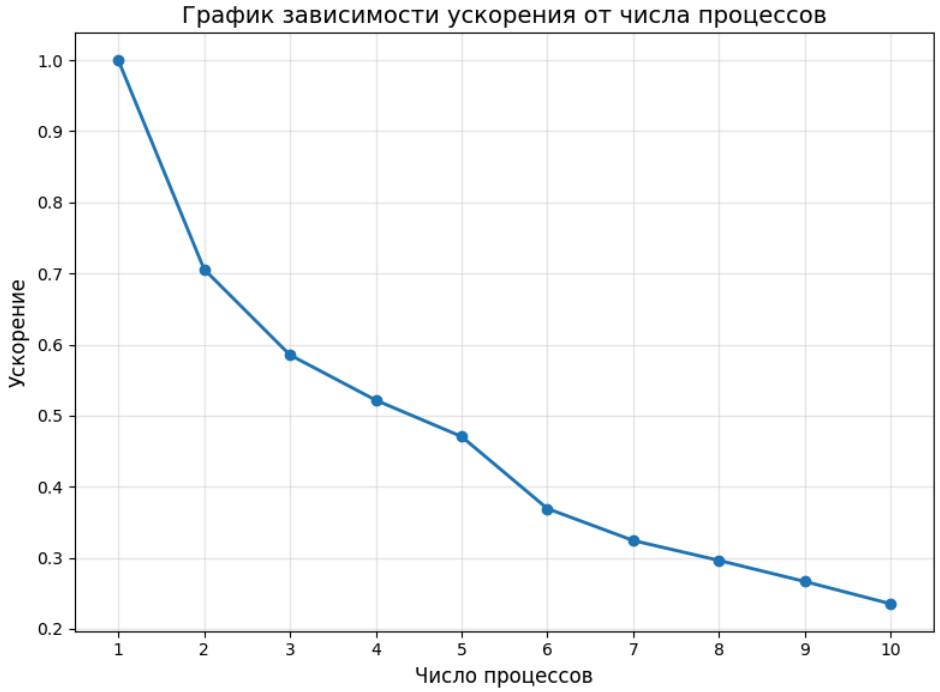

Рисунок 3 - График зависимости ускорения от числа процессов

Из полученных результатов можно сделать вывод, что нет определенной закономерности или порядка приема сообщений. В функции приема сообщений процесса с рангом 0 установлено *MPI_ANY_SOURCE*, что означает, что процесс

принимает все сообщения от любых процессов в порядке их поступления, следовательно, вследствие конкуренции процессов, обусловленной множеством факторов, результат каждый раз разный.

Для выполнения задания 2 отредактирован код программы для задания 1. Изменения коснулись правила приема сообщений. В предыдущей программе использован цикл от 0 до *num_proc-1*, и прием сообщений при помощи *MPI_ANY_SOURCE*, поэтому *num_proc-1* раз принимались сообщения от любых процессов. Теперь использован цикл от 1 до num_proc, а в функции приема выставлено текущее значение процесса, от которого нужно принять сообщение.

Сеть Петри для задания 2 представлена на рис.4.

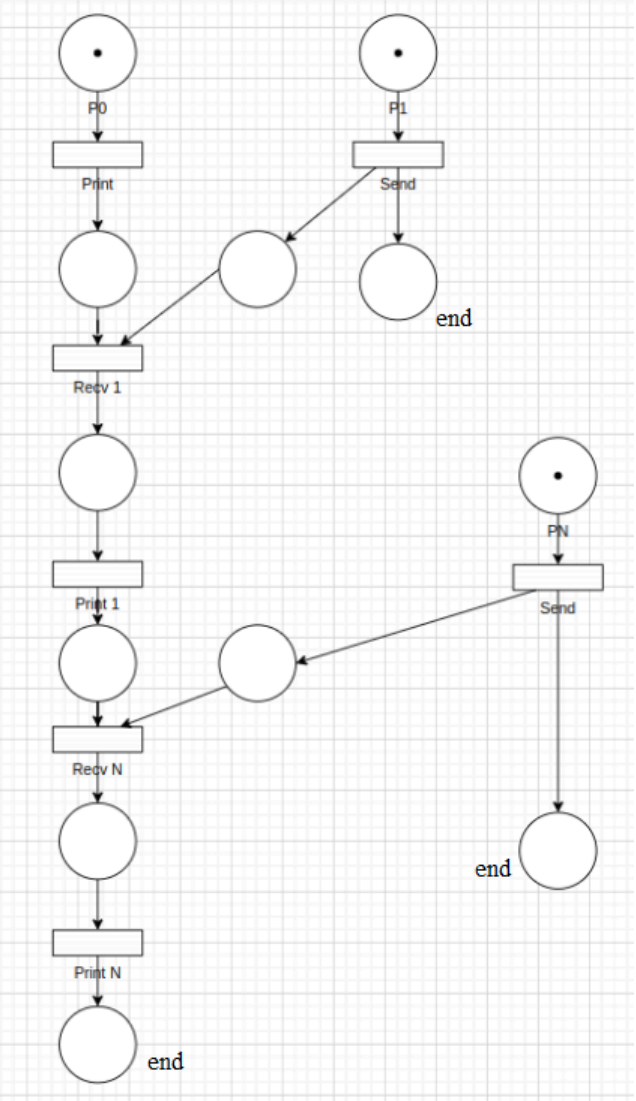

Рисунок 4 - Сеть Петри для задания 2

Код программы для задания 2, написанный на языке программирования C++ см. Приложение А.

Результаты работы программы см. Приложение В.

Зависимость времени работы программы для задания номер 2 от количества процессов можно увидеть в табл.2 и на рис.5.

Табл.2 см. Приложение Б.

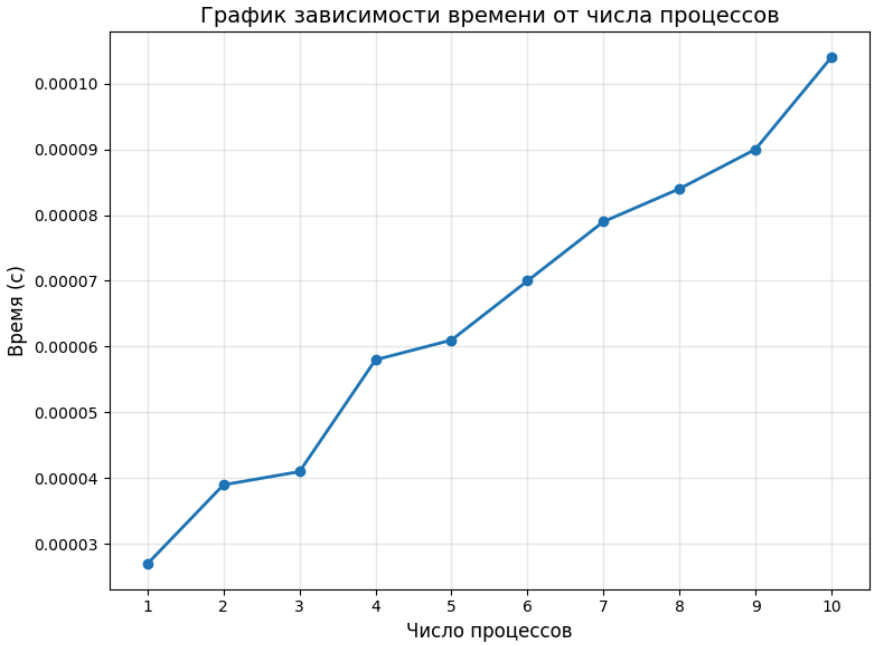

Рисунок 5- График зависимости времени от числа процессов

На рис.6 изображен график зависимости ускорения от числа процессов для задачи 2.

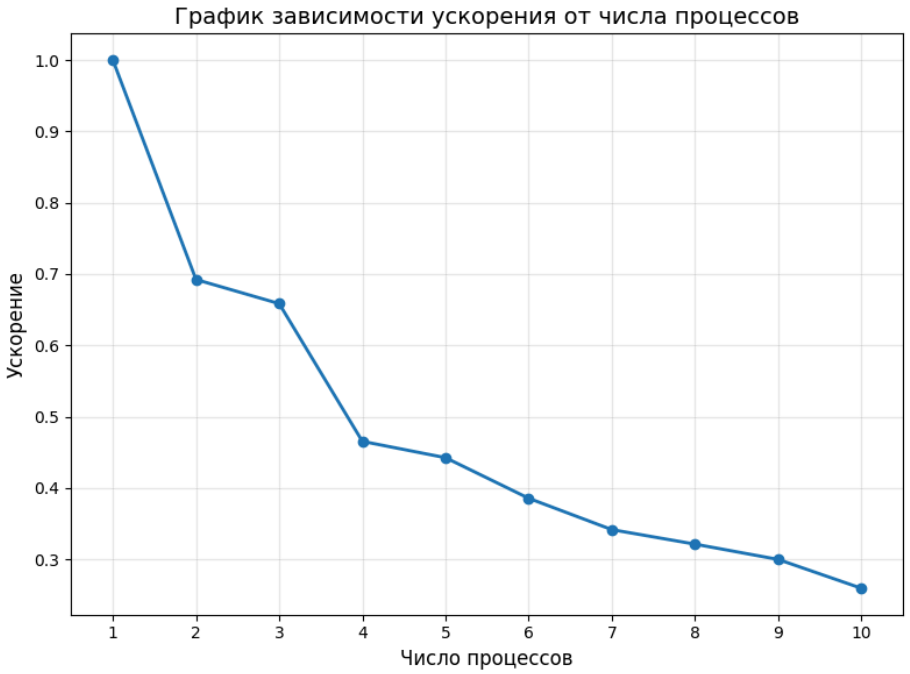

Рисунок 6 - График зависимости ускорения от числа процессов

Исходя из полученных результатов, время работы обеих программ линейно зависит от числа процессов, на которых они запущены. Что логично, так как при увеличении количества процессов на n, процессу с рангом 0 приходится делать на n итераций цикла больше. Кроме этого в обоих случаях с ростом количества процессов уменьшается ускорение, что обусловлено этими же причинами.

Выводы.

Изучены основы MPI, на основании которых написаны параллельные программы, запускаемые на различном числе одновременно работающих процессов. Проверена работоспособность программ на различном числе процессов, а также замерено время их работы и ускорение. Кроме этого выполнено упорядочивание приема сообщений, путем ожидания сообщений от конкретных процессов.

ПРИЛОЖЕНИЕ
ИСХОДНЫЙ КОД

Файл mpi_lb1_1.cpp:

```
#include <iostream>
#include <mpi.h>

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

   MPI_Init(&argc, &argv);

   int proc_rang, recv_rang, num_proc;
   MPI_Comm_rank(MPI_COMM_WORLD, &proc_rang);
   MPI_Comm_size(MPI_COMM_WORLD, &num_proc);

   if (proc_rang != 0){
      MPI_Send(&proc_rang, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
   }
   else {
      std::cout << "in work process with rang " << proc_rang << "\n";
      MPI_Status status;

      for (int i = 0; i < num_proc - 1; i++){
         MPI_Recv(&recv_rang, 1, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status);
         std::cout << "recieved message from process with rang " << recv_rang << "\n";
      }
   }
   MPI_Finalize();
}
```

Файл mpi_lb1_2.cpp:

```
#include <iostream>
#include <mpi.h>

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

   MPI_Init(&argc, &argv);

   int proc_rang, recv_rang, num_proc;
   MPI_Comm_rank(MPI_COMM_WORLD, &proc_rang);
   MPI_Comm_size(MPI_COMM_WORLD, &num_proc);

   if (proc_rang != 0){
```

```
        MPI_Send(&proc_rang, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
    }
    else {
        std::cout << "in work process with rang " << proc_rang << "\n";
        MPI_Status status;

        for (int i = 1; i < num_proc; i++){
            MPI_Recv(&recv_rang, 1, MPI_INT, i, 0, MPI_COMM_WORLD, &status);
            std::cout << "recieved message from process with rang " << recv_rang << "\n";
        }
    }
    MPI_Finalize();
}
```

ПРИЛОЖЕНИЕ Б
ТАБЛИЦЫ

Таблица 1 - Зависимость времени работы программы от числа процессов.

Количество процессов	Среднее время выполнения (с)
1	0.000024
2	0.000034
3	0.000041
4	0.000046
5	0.000051
6	0.000065
7	0.000074
8	0.000081
9	0.000090
10	0.000102

Таблица 2 - Зависимость времени работы программы от числа процессов

Количество процессов	Среднее время выполнения (с)
1	0.000027
2	0.000039
3	0.000041
4	0.000058
5	0.000061
6	0.000070
7	0.000079
8	0.000084
9	0.000090
10	0.000104

ПРИЛОЖЕНИЕ В
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

При нескольких запусков программы 1 получены подобные результаты:

in work process with rang 0

recieved message from process with rang 6

recieved message from process with rang 1

recieved message from process with rang 2

recieved message from process with rang 3

recieved message from process with rang 4

recieved message from process with rang 5

recieved message from process with rang 7

...

in work process with rang 0

recieved message from process with rang 4

recieved message from process with rang 3

recieved message from process with rang 7

recieved message from process with rang 1

recieved message from process with rang 2

recieved message from process with rang 5

recieved message from process with rang 6

При запуске программы 2 получен результат:

in work process with rang 0

recieved message from process with rang 1

recieved message from process with rang 2

recieved message from process with rang 3

recieved message from process with rang 4

recieved message from process with rang 5

recieved message from process with rang 6

recieved message from process with rang 7