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

Студент гр. 9381 _____ Птичкин С. А.
Преподаватель _____ Татаринов Ю. С.

Санкт-Петербург
2021

Задание.

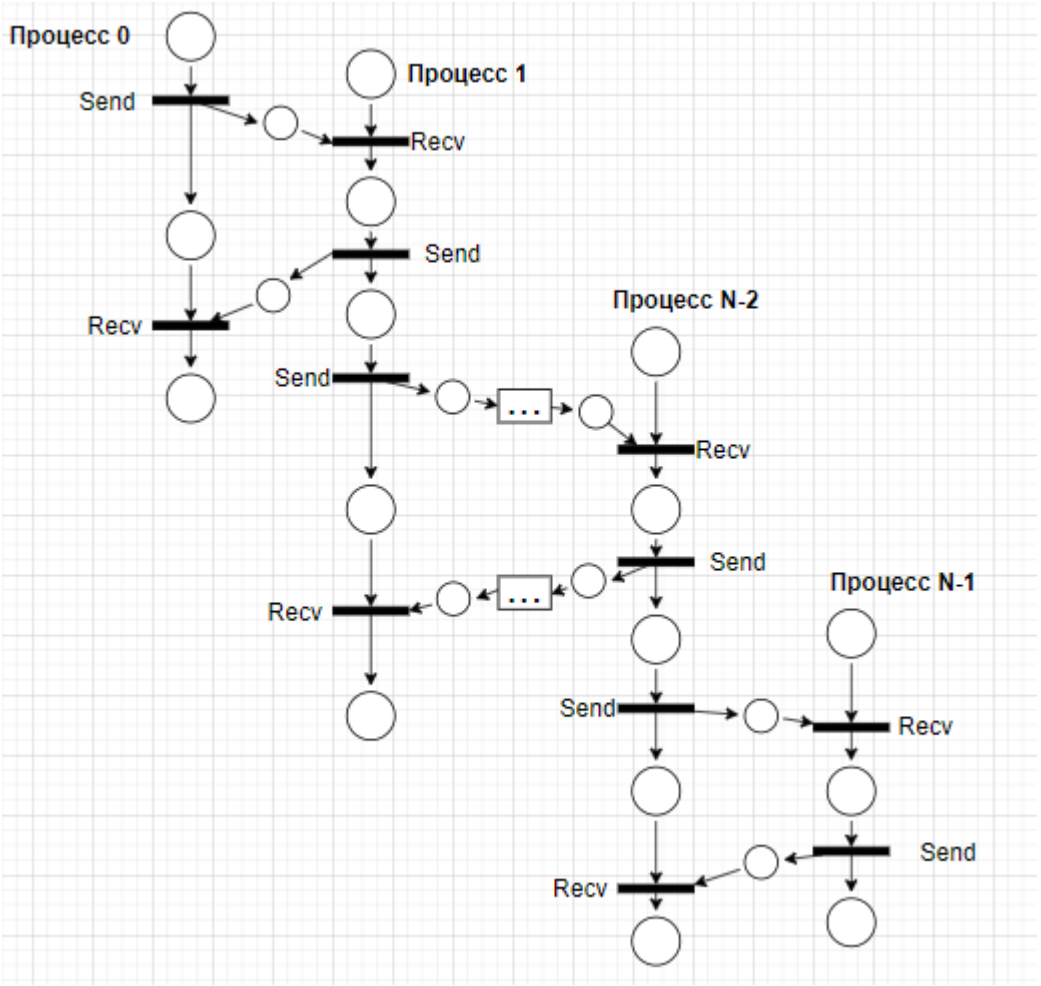
Вариант № 11.

Разговор. Несколько процессов обмениваются между собой сообщениями, требующими ответа источнику сообщений. Тем самым моделируется разговор между несколькими людьми.

Краткое описание алгоритма.

Обмен процессов сообщениями происходит следующим образом: процесс с индексом 0 начинает передачу сообщению 1, тот принимает его и отправляет ответ. Затем процесс 1 начинает обмен с следующим по порядку процессом, ожидая от него ответа. Заканчивается цепочка передач на процессе с индексом N-1, где N - число процессов, на которых запущена программа. Так как для передачи нужно как минимум 2 процесса, запуск на 1 процессе не имеет смысла и в исследованиях не учитывается.

Формальное описание алгоритма.



Листинг программы.

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

int main(int argc, char* argv[]) {
      int ProcNum, ProcRank;
      char send_message[] = "Hello";
      int len = sizeof(send_message) / sizeof(char);
      char* rec_message = new char[len];

      MPI_Status Status;
      MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD, &ProcNum);
      MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);
      if (ProcRank == 0) {
            MPI_Send(&send_message,     len,     MPI_CHAR,     1,     0,
MPI_COMM_WORLD);
            MPI_Recv(rec_message, len, MPI_CHAR, 1, 0, MPI_COMM_WORLD,
```

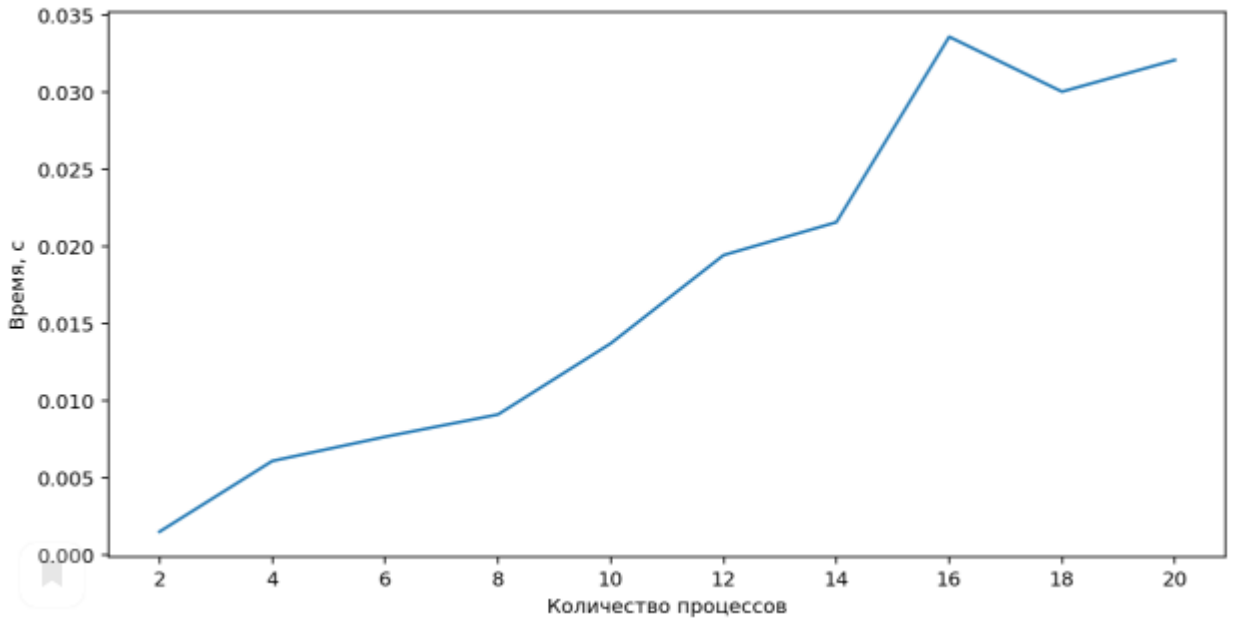
```
&Status);
          printf("Process %d received msg from process %d: %s",
ProcRank, 1, rec_message);
     }
     else if (ProcRank == ProcNum - 1) {
          MPI_Recv(rec_message, len, MPI_CHAR, ProcRank - 1, 0,
MPI_COMM_WORLD, &Status);
          printf("Process %d received msg from process %d: %s",
ProcRank, ProcRank - 1, rec_message);
          MPI_Send(&send_message, len, MPI_CHAR, ProcRank - 1, 0,
MPI_COMM_WORLD);
     }
     else {
          MPI_Recv(rec_message, len, MPI_CHAR, ProcRank - 1, 0,
MPI_COMM_WORLD, &Status);
          printf("Process %d received msg from process %d: %s",
ProcRank, ProcRank-1, rec_message);
          MPI_Send(&send_message, len, MPI_CHAR, ProcRank - 1, 0,
MPI_COMM_WORLD);
          MPI_Send(&send_message, len, MPI_CHAR, ProcRank + 1, 0,
MPI_COMM_WORLD);
          MPI_Recv(rec_message, len, MPI_CHAR, ProcRank + 1, 0,
MPI_COMM_WORLD, &Status);
     }

     MPI_Finalize();
     delete[] rec_message;
     return 0;
}
```

Выполнение работы.

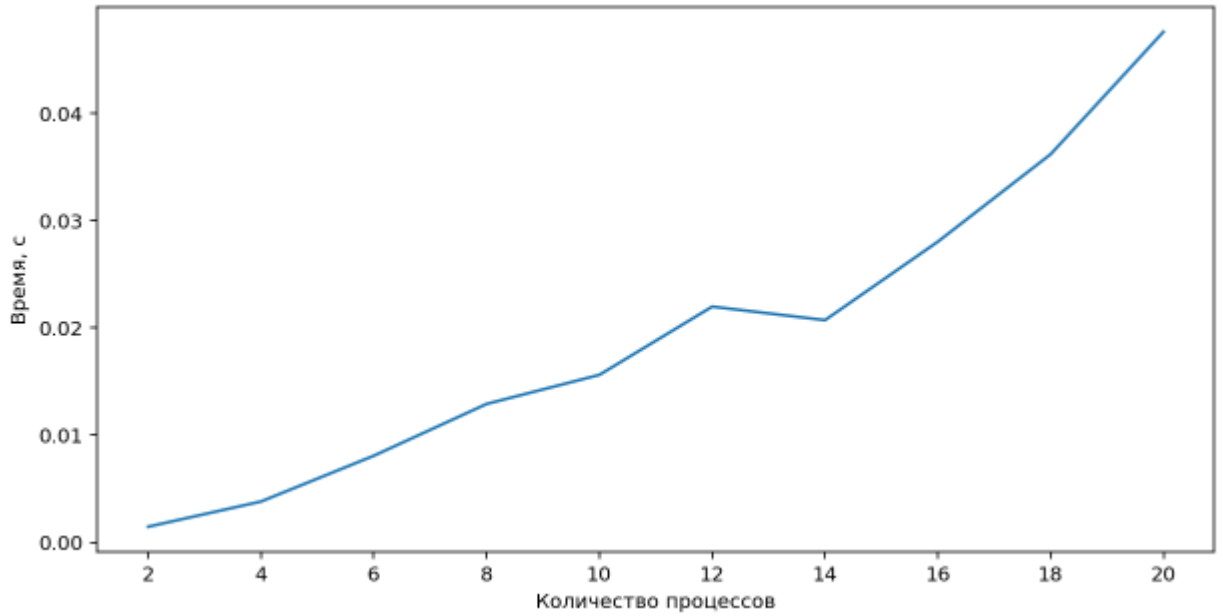
1) Была написана программа обмена сообщениями между несколькими процессами. Затем она была протестирована на различном числе процессов:

```
C:\Users\Сергей\Documents\Visual Studio 2015\Projects\par_alg_1\Debug>mpiexec -n 2 par_alg_1.exe
Process 0 received msg from process 1: Hello
Process 1 received msg from process 0: Hello
C:\Users\Сергей\Documents\Visual Studio 2015\Projects\par_alg_1\Debug>mpiexec -n 3 par_alg_1.exe
Process 2 received msg from process 1: Hello
Process 0 received msg from process 1: Hello
Process 1 received msg from process 0: Hello
C:\Users\Сергей\Documents\Visual Studio 2015\Projects\par_alg_1\Debug>mpiexec -n 6 par_alg_1.exe
Process 2 received msg from process 1: Hello
Process 0 received msg from process 1: Hello
Process 1 received msg from process 0: Hello
Process 5 received msg from process 4: Hello
Process 3 received msg from process 2: Hello
Process 4 received msg from process 3: Hello
C:\Users\Сергей\Documents\Visual Studio 2015\Projects\par_alg_1\Debug>mpiexec -n 10 par_alg_1.exe
Process 3 received msg from process 2: Hello
Process 4 received msg from process 3: Hello
Process 2 received msg from process 1: Hello
Process 1 received msg from process 0: Hello
Process 5 received msg from process 4: Hello
Process 0 received msg from process 1: Hello
Process 6 received msg from process 5: Hello
Process 7 received msg from process 6: Hello
Process 9 received msg from process 8: Hello
Process 8 received msg from process 7: Hello
```

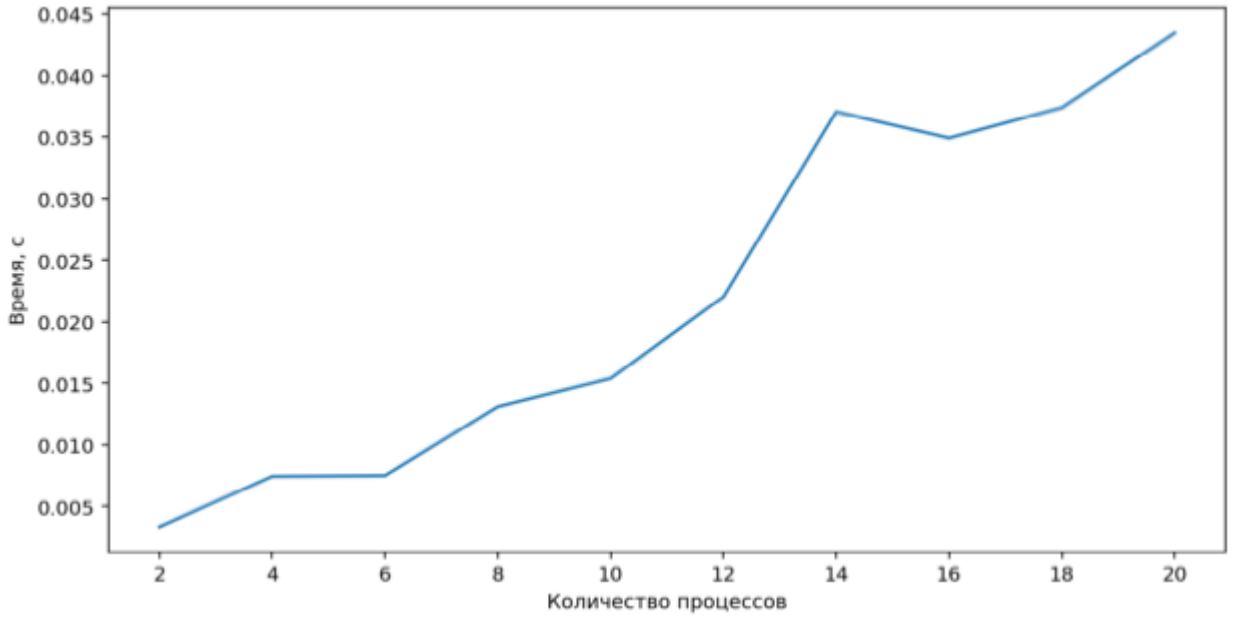
2) Были построены графики зависимости времени выполнения программы от числа процессов для разных длин пересылаемых сообщений. Длина сообщения 10000 символов:



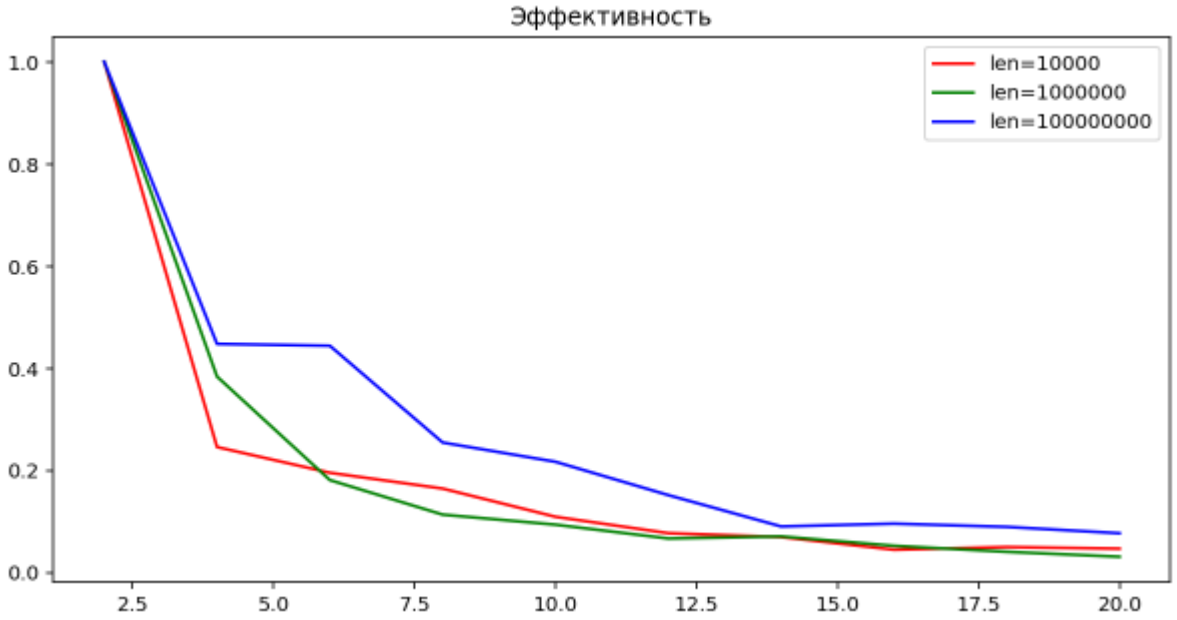
Длина сообщения 1000000 символов:



Длина сообщения 100000000 символов:



3) Был построен график ускорения программы:



Вывод.

Таким образом, была написана программа по обмену сообщениями между несколькими процессами, требующими ответа. Произведены измерения времени выполнения программы на различном числе процессов при разной длине передаваемого сообщения. Также построен график ускорения программы, исследовав который можно сделать вывод, что при увеличении количества процессов программа работает медленнее, что логично, так как число передач между процессами увеличивается с их количеством.