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 и упорядочение вывода ее результатов

Студент гр. 9381 _____ Колованов Р.А.

Преподаватель _____ Татаринов Ю.С.

Санкт-Петербург
2021

Цель работы.

Запуск параллельной программы на различном числе одновременно работающих процессов, упорядочение вывода результатов.

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.

Выполнение работы.

Для начала программа была скомпилирована и запущена с разным количеством процессов несколько раз. Были выделены следующие правила, по которым сообщения выводятся на экран:

- В самом начале всегда выводится сообщение с номером процесса 0;
- После этого выводится N – 1 сообщений со случайным порядком следования номеров.

Далее программа был модифицирована таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего процесса.

Детерминированность результатов работы параллельного алгоритма уменьшает эффективность алгоритма, поскольку в этом случае требуется внедрение механизмов синхронизации, которые замедляют скорость работы.

Краткое описание модификации программы.

Для приема сообщений от процессов по порядку от 1 до N – 1 в функции отправки и приема вместо MPI_ANY_SOURCE был передан номер процесса, от которого мы хотим получить сообщение. Благодаря этой модификации процессор 0 будет принимать сообщения по порядку от других процессов с номерами от 1 до N – 1.

Листинг программы.

Листинг 1. Программа до модификации.

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


int main(int argc, char** argv) {
      int processNumber, processRank, recievedRank;
      MPI_Status status;

      MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD, &processNumber);
      MPI_Comm_rank(MPI_COMM_WORLD, &processRank);

      if (processRank == 0) {
            std::cout << "Hello from process " << processRank << "\n";

            for (int i = 1; i < processNumber; i++) {
                  MPI_Recv(&recievedRank,   1,   MPI_INT,   i,   MPI_ANY_TAG,
MPI_COMM_WORLD, &status);
                  std::cout << "Hello from process " << recievedRank << "\n";
            }

      } else {
            MPI_Send(&processRank, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
      }

      MPI_Finalize();

      return 0;
}
```

Листинг 2. Программа после модификации.

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


int main(int argc, char** argv) {
      int processNumber, processRank, recievedRank;
      MPI_Status status;

      MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD, &processNumber);
      MPI_Comm_rank(MPI_COMM_WORLD, &processRank);

      if (processRank == 0) {
            std::cout << "Hello from process " << processRank << "\n";

            for (int i = 1; i < processNumber; i++) {
                  MPI_Recv(&recievedRank,   1,   MPI_INT,   MPI_ANY_SOURCE,   i,
MPI_COMM_WORLD, &status);
                  std::cout << "Hello from process " << recievedRank << "\n";
            }

      } else {
            MPI_Send(&processRank, 1, MPI_INT, 0, processRank, MPI_COMM_WORLD);
      }

      MPI_Finalize();
```

```
    return 0;
}
```

Результаты работы программы.

До модификации:

№ п/п	Количество процессоров	Результаты работы программы
1.	1	Hello from process 0
2.	2	Hello from process 0 Hello from process 1
3.	4	Hello from process 0 Hello from process 2 Hello from process 1 Hello from process 3
4.	4	Hello from process 0 Hello from process 1 Hello from process 3 Hello from process 2
5.	8	Hello from process 0 Hello from process 6 Hello from process 4 Hello from process 3 Hello from process 5 Hello from process 7 Hello from process 1 Hello from process 2
6.	8	Hello from process 0 Hello from process 6 Hello from process 4 Hello from process 2 Hello from process 1 Hello from process 5 Hello from process 3 Hello from process 7

После модификации:

№ п/п	Количество процессоров	Результаты работы программы
1.	1	Hello from process 0
2.	2	Hello from process 0 Hello from process 1
3.	4	Hello from process 0 Hello from process 1

		Hello from process 2 Hello from process 3
5.	8	Hello from process 0 Hello from process 1 Hello from process 2 Hello from process 3 Hello from process 4 Hello from process 5 Hello from process 6 Hello from process 7

Выводы.

Были получены навыки компиляции, разработки и запуска параллельной программы, использующей библиотеку MPI, на различном числе одновременно работающих процессов. Была выполнена модификация программы таким образом, чтобы порядок вывода сообщений на экран соответствовал номеру соответствующего процесса.

Детерминированность результатов работы параллельного алгоритма уменьшает эффективность алгоритма, поскольку в этом случае требуется внедрение механизмов синхронизации, которые замедляют скорость работы.