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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Параллельные алгоритмы»

Тема: Использование аргументов - джокеров.

Студент гр. 1384 Усачева Д. В.

Преподаватель Татаринов Ю. С.

Санкт-Петербург

2023

Цель

Цель данной работы заключается в разработке и реализации параллельной программы, использующей аргументы-джокеры.

Задание

Варианта №5.

Игра в снежки. Процессы делятся на 2 группы (четные и нечетные номера). Каждый процесс нечетной группы случайным образом генерирует посылку четному процессу. Четный процесс, получив посылку, в свою очередь, отправляет случайно выбранному нечетному процессу следующую. Предусмотреть ситуацию получения одним процессом нескольких посылок.

Выполнение работы

Для выполнения поставленной задачи написана программа на языке C++, код которой представлен ниже в листинге 1.

В цикле с ограничением по времени происходит отправка и получение посылок. Если процесс имеет четный ранг, то сначала происходит проверка наличия сообщения при помощи функции MPI_Iprobe, далее если сообщение доступно для приема, процесс принимает его и выводит информацию, содержащую ранг получателя и отправителя. Если же сообщения от других процессов нет, то происходит отправка сообщения данным процессом случайному нечетному процессу. Для процессов нечетного ранга сначала происходит отправка сообщения случайному процессу четного ранга, а потом проверка и получение сообщений от других процессов.

Для получения сообщений процессами используется функция MPI_Recv с джокером MPI_ANY_SOURCE, чтобы процесс мог принять сообщение от отправителя с любым рангом. Для отправки сообщений процессами время ограничено и в несколько раз меньше ограничения в основном цикле. Это необходимо, так как получение сообщения происходит только при его наличии. Времени на получение может понадобиться гораздо больше, так как один процесс может получить сразу несколько сообщений от процессов из другой группы. Такие ограничения позволяют процессам

избежать блокировки из-за того, что все сообщения будут получены, и корректно завершить программу.

Ниже представлена сеть Петри основной части алгоритма (см. рис 1).

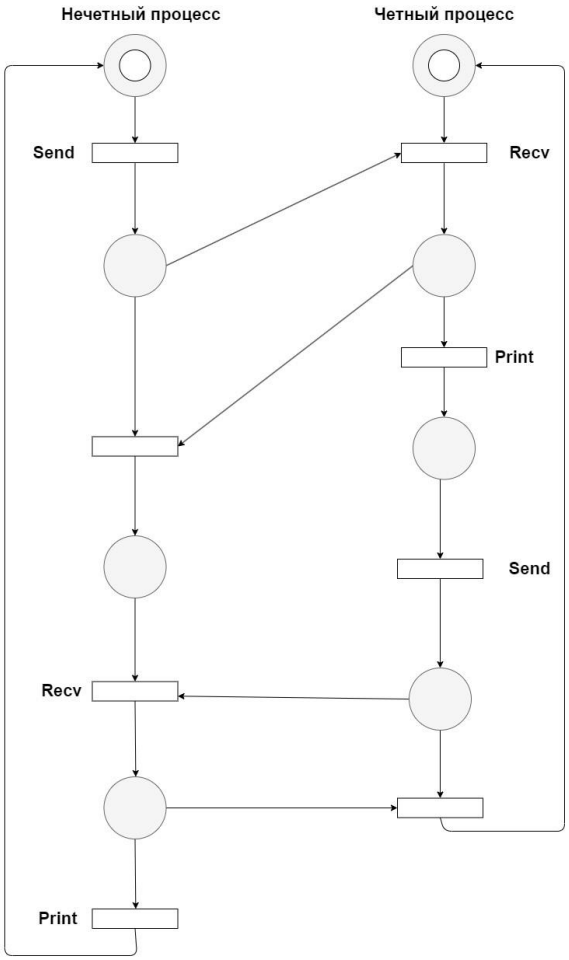

Рисунок 1 — Сеть Петри основной части алгоритма

Листинг 1 — Код программы lab2.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <random>

int main(int argc, char** argv) {
    int procNum, procRank;
    double start;
    int received_message;
    int flag = 0;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);
    MPI_Comm_size(MPI_COMM_WORLD, &procNum);
    start = MPI_Wtime();
    while(MPI_Wtime() - start < 1){
        if (procRank % 2 == 0){
            MPI_Iprobe(MPI_ANY_SOURCE,                MPI_ANY_TAG,
MPI_COMM_WORLD, &flag, MPI_STATUS_IGNORE);
```

```
            if(flag){

MPI_Recv(&received_message,1,MPI_INT,MPI_ANY_SOURCE,0,MPI_COMM_WO
RLD,MPI_STATUS_IGNORE);
                printf("Process %d received a message from process
%d\n",procRank,received_message);
            }
            int address = 0;
            while(address % 2 == 0){
                std::random_device device;
                address = device() % procNum;
            }
            if (MPI_Wtime() - start < 0.0002){

MPI_Send(&procRank,1,MPI_INT,address,0,MPI_COMM_WORLD);
            }
        }
        else{
            int address = 1;
            while(address % 2 == 1){
                std::random_device device;
                address = device() % procNum;
            }
            if (MPI_Wtime() - start < 0.0002){

MPI_Send(&procRank,1,MPI_INT,address,0,MPI_COMM_WORLD);
            }
            MPI_Iprobe(MPI_ANY_SOURCE,                MPI_ANY_TAG,
MPI_COMM_WORLD, &flag, MPI_STATUS_IGNORE);
            if(flag){

MPI_Recv(&received_message,1,MPI_INT,MPI_ANY_SOURCE,0,MPI_COMM_WO
RLD,MPI_STATUS_IGNORE);
                printf("Process %d received a message from process
%d\n",procRank,received_message);
            }
        }
    }
    printf("Process %d finished \n",procRank);
    MPI_Finalize();
    return 0;
}
```

Ниже представлен вывод программы lab2.cpp

Листинг 2 — Вывод программы lab2.cpp для 3 процессов

```
Process 0 received a message from process 1
Process 1 received a message from process 0
Process 0 received a message from process 1
Process 1 received a message from process 2
Process 1 received a message from process 0
Process 2 received a message from process 1
Process 1 received a message from process 2
Process 0 received a message from process 1
Process 1 received a message from process 0
Process 1 received a message from process 2
Process 1 received a message from process 0
Process 1 received a message from process 2
```

```
Process 1 finished
Process 0 finished
Process 2 finished
```

Время работы программы не зависит от объема отправляемого сообщения и количества процессом, оно одинаково для всех процессов и равно ограничению в основном цикле программы. Следовательно, график зависимости времени от количества процессов прямая-константа, а график ускорения прямая-единица.

Выводы

В ходе выполнения лабораторной работы была реализована параллельная программа, использующая аргументы-джокеры. Выполнена пересылка сообщений между двумя группами процессов.

Выявлено, что время работы данной программы статично и не зависит от объема сообщения и количества процессов.