**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНСТИТУТ ЦИФРЫ**

**ДОМАШНЯЯ РАБОТА № 2**

**ПО ДИСЦИПЛИНЕ**

**“Технологии параллельных вычислений”**

**Задача № 6**

студентки 3 курса

**Колесник Полины Олеговны**

Направление 02.03.02 – Фундаментальная информатика и информационные технологии

Преподаватель:

к-т физ.-мат.наук, доцент

С.В. Стуколов

Работа защищена

« »

“ ” 2023 г.

Кемерово 2023 г.

СОДЕРЖАНИЕ

[1. Постановка задачи 3](#_Toc145840870)

[2. Описание используемых функций 3](#_Toc145840871)

[3. Описание программы 3](#_Toc145840872)

[4. Реализация 3](#_Toc145840873)

[Заключение 6](#_Toc145840874)

[Литература 6](#_Toc145840875)

# 1. Постановка задачи

Напишите программу, используя блокирующие коммуникационные функции (MPI\_Send, MPI\_Recv), реализующую алгоритм передачи данных по кольцу. Алгоритм программы можно представить следующим образом:

* на 0 процессе инициализируется переменная (int a);
* 0 процесс пересылает переменную а первому процессу;
* первый, получив значение переменной а, добавляет к нему единицу и отправляет на 2-й процесс.
* …
* (size-1) процесс, получив значение переменной а от (size-2) процесса, прибавляет к значению а единицу и отправляет на 0-й процесс.

# 2. Описание используемых функций

# 3. Описание программы

Реализация алгоритма передачи данных по кольцу.

# 4. Реализация

Текст программы:

#include <stdio.h>

#include "mpi.h"

#include "iostream"

using namespace std;

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

{

    int rank, size, next, prev;

    int a = 0;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    next = rank + 1;

    prev = rank - 1;

    if(rank == 0) {

        prev = size - 1;

    }

    if(rank == size - 1)  {

        next = 0;

    }

    if(rank==0)

    {

    cout<<"Enter a: ";

        cin>>a;

        MPI\_Send(&a,1,MPI\_INT,next,777,MPI\_COMM\_WORLD);

        MPI\_Recv(&a,1,MPI\_INT,prev,777,MPI\_COMM\_WORLD, &stat);

    }

    else

    {

        MPI\_Recv(&a,1,MPI\_INT,prev,777,MPI\_COMM\_WORLD,&stat);

        a++;

        MPI\_Send(&a,1,MPI\_INT,next,777,MPI\_COMM\_WORLD);

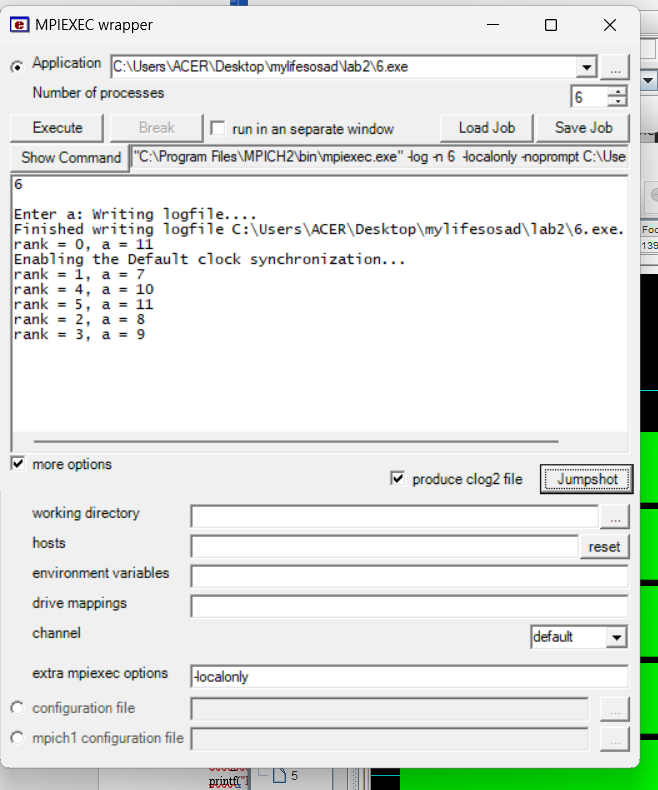
    }

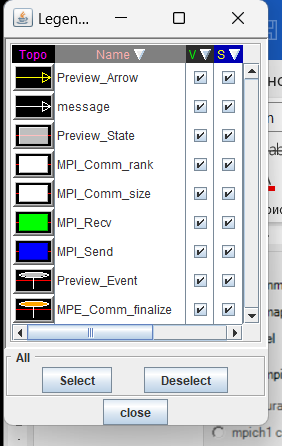
    printf( "rank = %d, a = %d \n",rank, a );

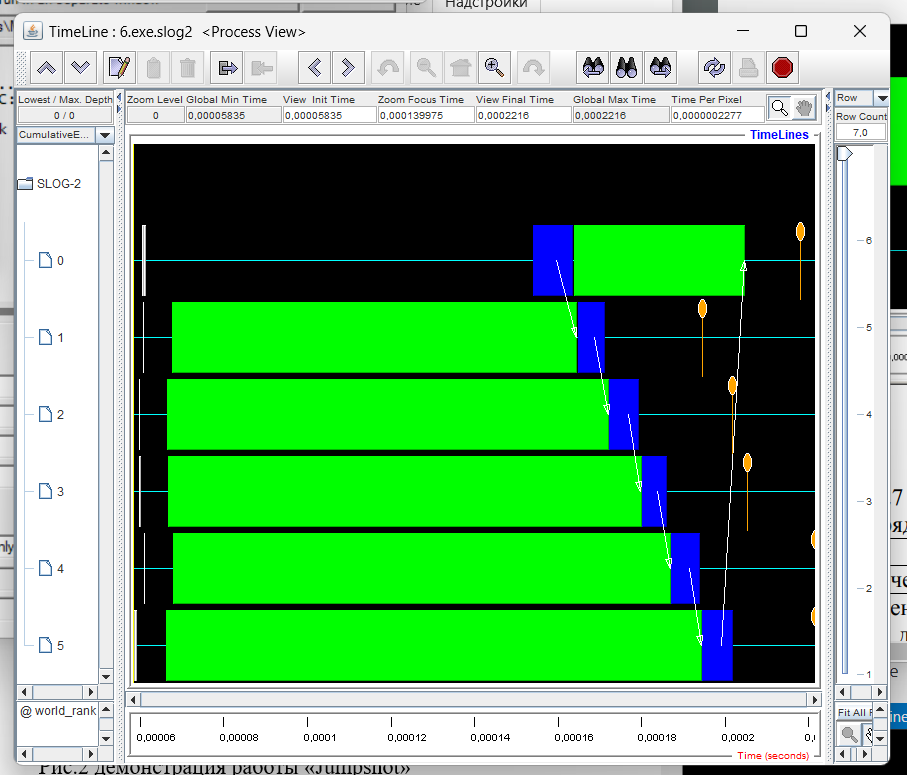
    MPI\_Finalize();

    return 0;

}

  
Рис.1 результат работы

  
Рис.2 демонстрация работы «Jumpshot»

  
 Рис.3 демонстрация работы «Jumpshot»

На трассе видно, что программа запущена на 6 процессах, 0-ой процесс отправляет данные на первый процесс и переходит в режим ожидания получения результата от последнего процесса. Пока 0-ой отправлял данные, остальные ожидали. Когда первый получил число, он отправляет результат следующему и так до size-1. Когда результат пришёл последнему, он заново отправляет данные 0-му

# Заключение

# Литература