**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНСТИТУТ ЦИФРЫ**

**ДОМАШНЯЯ РАБОТА № 9**

**ПО ДИСЦИПЛИНЕ**

**“Технологии параллельных вычислений”**

**Задача № 1**

студенток 3 курса

**Колесник Полины Олеговны, Малоштановой Ольги Алексеевны**

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

# 1. Постановка задачи

Используя коллективные коммуникационные функции, создайте следующую параллельную программу: на всех процессах задается одномерный массив (a[i]=rank, i=0...2), который отправляется на 0-й процесс; 0-й процесс принимает от всех остальных пересылаемые данные в одномерный массив. Например, Вы запускаете программу на 3-х процессах, 1-й процесс отправляет 0-му следующий массив (1,1,1), 2-й процесс отправляет 0-му - (2,2,2), 0-й процесс получает данные и, сохраняя в одномерный массив, выводит на экран следующее: 1,1,1,2,2,2.

# 2. Описание используемых функций

# 3. Описание программы

# 4. Реализация

Текст программы:

# Заключение

# Литература

**№2**

# 1. Постановка задачи

Создайте следующую параллельную программу: на всех процессах задается одномерный массив (a[i]=rank, i=0...size-rank, который отправляется на 0-й процесс; 0-й процесс принимает от всех остальных пересылаемые данные. Например, Вы запускаете программу на 4-х процессах, 0-й процесс будет отправлять следующий массив (0,0,0…0), состоящий из size элементов, 1-й процесс отправляет 0-му следующий массив (1,1…1), состоящий из size-1 элемента, …, (size-1)-й процесс отправляет 0-му - (size-1), 0-й процесс получает данные и, сохраняя в одномерный массив, выводит на экран следующее: 0,…,0,1…1,…,(size-1).

# 2. Описание используемых функций

MPI\_Gatherv - позволяет собирать блоки с разным числом элементов от каждого процесса.

# 3. Описание программы

На всех процессах задаётся массив и отправляются на 0, а 0 процесс принимает данные от остальных.

# 4. Реализация

Текст программы:

#include <stdio.h>

#include "mpi.h"

#include "iostream"

using namespace std;

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

{

    int rank;

    int size;

    int n = 0;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    int \*b = new int[size - rank];

    for (int i = 0; i < size - rank; i++)

        b[i] = rank;

    int \*a = new int[size \* (size + 1) / 2];

    int \*bl = new int[size];

    int \*ds = new int[size];

    ds[0] = 0;

    for (int i = 1; i < size; i++)

    {

        ds[i] = ds[i - 1] + size - i + 1;

    }

    for (int i = 0; i < size; i++)

    {

        bl[i] = size - i;

    }

    MPI\_Gatherv(b, size - rank, MPI\_INT, a, bl, ds, MPI\_INT, 0, MPI\_COMM\_WORLD);

    double s = 0;

    for (int i = 0; i < rank \* 1000000; i++)

        s += 0.0000000001;

    if (rank == 0)

    {

        printf("rank = %d, a: ", rank);

        for (int i = 0; i < size \* (size + 1) / 2; i++)

            cout << a[i] << " ";

    }

    // printf( "rank = %d, a: ",rank );

    cout << endl;

    delete[] a;

    delete[] b;

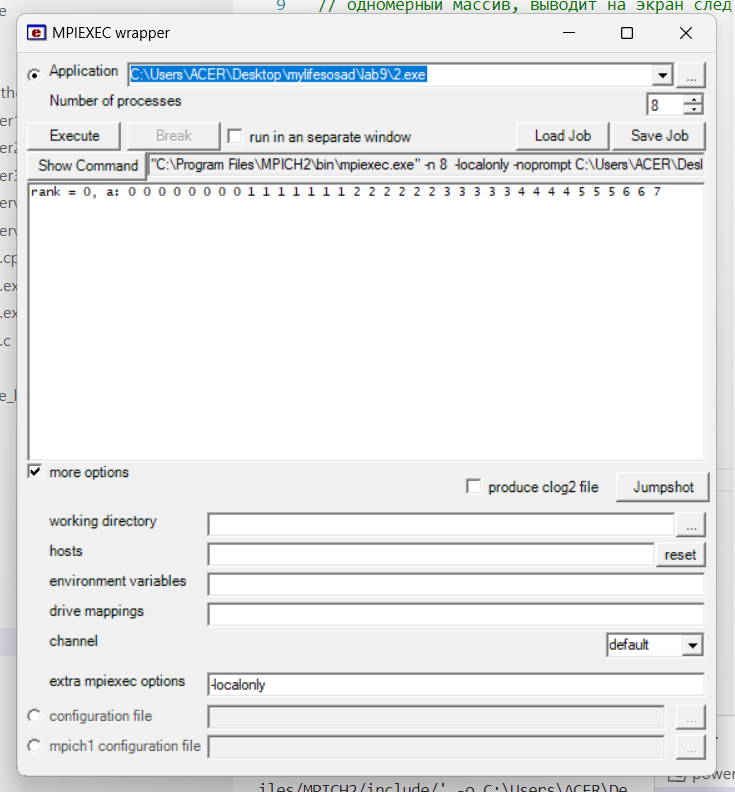
    delete[] bl;

    delete[] ds;

    MPI\_Finalize();

    return 0;

}



# Заключение

# Литература

**№3**

# 1. Постановка задачи

Напишите программу, которая реализует сборку по две строки с каждого процесса в двумерный массив на 0-м процессе по следующему алгоритму: на 0-м процессе считывается значение переменной n и рассылается каждому; на каждом процессе выделяется место в памяти под двумерный массив a[2][n], который заполняется номером процесса; на всех процессах выделяется место в памяти под двумерный массив b[2\*size][n], после чего производится сборка массивов a в массив b на 0-м процессе.

# 2. Описание используемых функций

# 3. Описание программы

# 4. Реализация

Текст программы:

# Заключение

# Литература

**№4**

# 1. Постановка задачи

На основе примера 6 напишите программу, которая собирает с каждого процесса по две строки длиной (rank+1) элементов.

# 2. Описание используемых функций

MPI\_Gatherv - позволяет собирать блоки с разным числом элементов от каждого процесса.

# 3. Описание программы

На всех процессах задается двумерный массив (b[i][j]), которые состоят из двух строчек и длиной rank+1, затем отправляются на 0-й процесс;

# 4. Реализация

Текст программы:

#include <stdio.h>

#include "mpi.h"

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

{

    int rank;

    int size;

    int i, s = 0, j;

    MPI\_Init(&argc, &argv);

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

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

    int \*\*b = new int \*[2];

    b[0] = new int[2 \* size];

    for (i = 1; i < 2; i++)

        b[i] = b[i - 1] + size;

    for (i = 0; i < 2; i++)

    {

        for (j = 0; j < rank + 1; j++)

            b[i][j] = rank;

        for (j = rank + 1; j < size; j++)

            b[i][j] = 0;

    }

    int \*\*a = new int \*[2 \* size];

    a[0] = new int[2 \* size \* size];

    for (i = 1; i < 2 \* size; i++)

        a[i] = a[i - 1] + size;

    int \*RC = new int[size];

    for (int i = 0; i < size; i++)

        RC[i] = 2 \* size;

    int \*ds = new int[size];

    for (int i = 0; i < size; i++)

        ds[i] = 2 \* i \* size;

    MPI\_Gatherv(\*b, 2 \* size, MPI\_INT, \*a, RC, ds, MPI\_INT, 0, MPI\_COMM\_WORLD);

    if (rank == 0)

    {

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

        for (i = 0; i < 2 \* size; i++)

        {

            for (j = 0; j < size; j++)

                printf(" %d ", a[i][j]);

            printf("\n ");

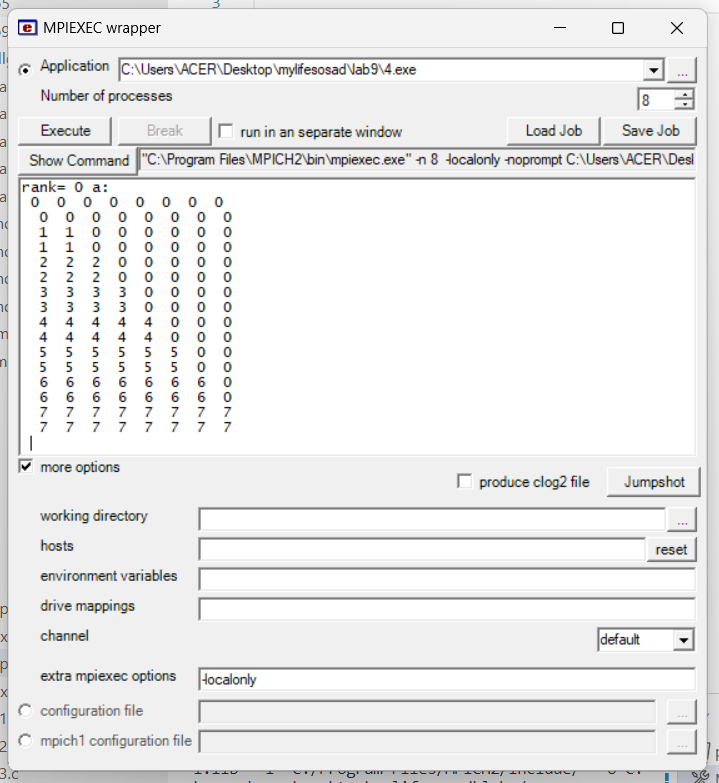
        }

    }

    MPI\_Finalize();

    return 0;

}



# Заключение

# Литература

**№5**

# 1. Постановка задачи

На основе примера 6 напишите программу, которая собирает с каждого процесса по одной строке, но с 0-го процесса – строку длины size элементов, с 1-го процесса – (size-1) элемент и т.д. Например, запустив программу на 5-и процессах, в итоге Вы должны получить следующую матрицу:

0 0 0 0 0

1 1 1 1 0

2 2 2 0 0

3 3 0 0 0

4 0 0 0 0

# 2. Описание используемых функций

# 3. Описание программы

# 4. Реализация

Текст программы:

# Заключение

# Литература

**№6**

# 1. Постановка задачи

Модифицируйте предыдущий пример, осуществляя сохранение каждой строки, начиная с элемента на главной диагонали. Например, запустив программу на 5-и процессах, в итоге Вы должны получить следующую матрицу:

0 0 0 0 0

0 1 1 1 1

0 0 2 2 2

0 0 0 3 3

0 0 0 0 4

# 2. Описание используемых функций

MPI\_Gatherv - позволяет собирать блоки с разным числом элементов от каждого процесса.

# 3. Описание программы

На каждом процессе создаётся массив, значения которых находятся на главной диагонали и выше, а потом посылаются на 0 процесс, а там они уже принимаются;

# 4. Реализация

Текст программы:

#include <stdio.h>

#include "mpi.h"

#include "iostream"

using namespace std;

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

{

    int rank;

    int size;

    int n = 0, i, j;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    int \*b = new int[size - rank];

    for (i = 0; i < size - rank; i++)

        b[i] = rank;

    int \*\*a = new int \*[size];

    a[0] = new int[size \* size];

    for (i = 1; i < size; i++)

        a[i] = a[i - 1] + size;

    for (i = 0; i < size; i++)

        for (j = 0; j < size; j++)

            a[i][j] = 0;

    int \*bl = new int[size];

    int \*ds = new int[size];

    for (int i = 0; i < size; i++)

    {

        bl[size - i - 1] = i + 1;

        ds[i] = i \* size + i;

    }

    MPI\_Gatherv(b, size - rank, MPI\_INT, \*a, bl, ds, MPI\_INT, 0, MPI\_COMM\_WORLD);

    if (rank == 0)

    {

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

        for (int i = 0; i < size; i++)

        {

            for (int j = 0; j < size; j++)

                cout << a[i][j] << " ";

            cout << endl;

        }

    }

    delete[] a[0];

    delete[] a;

    delete[] b;

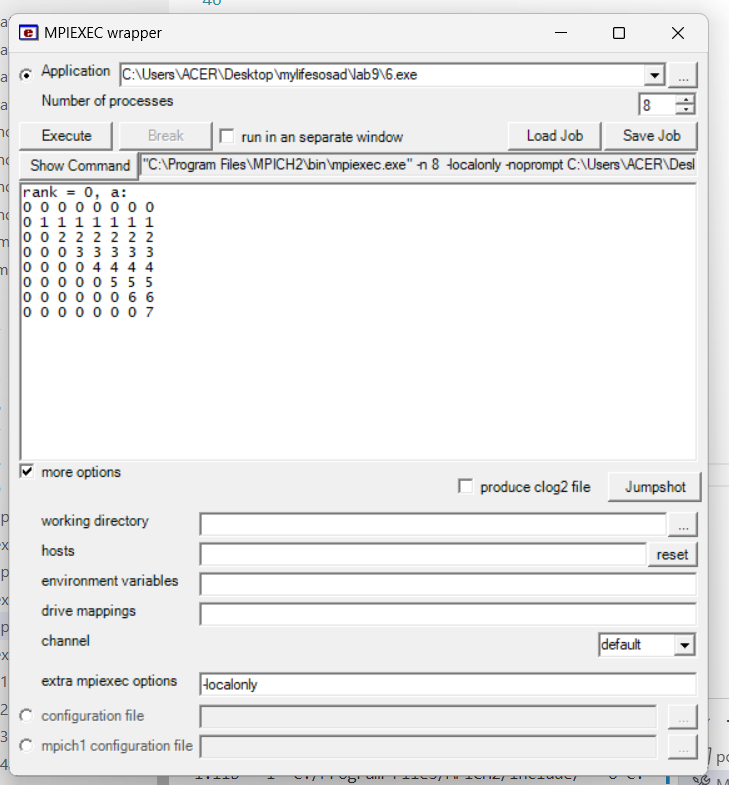
    delete[] bl;

    delete[] ds;

    MPI\_Finalize();

    return 0;

}



# Заключение

# Литература