МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
 «Кемеровский государственный университет»**

**Институт фундаментальных наук**

**ДОМАШНЯЯ РАБОТА №9**

**ПО ДИСЦИПЛИНЕ**

**“Технологии параллельных вычислений”**

студента 3 курса

**Сулима Роман Иванович**

Направление 02.03.02 – Фундаментальная информатика и информационные технологии

Преподаватель:

к-т физ.-мат.наук, доцент

С.В. Стуколов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена:

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_г.

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Кемерово 2021

СОДЕРЖАНИЕ

[1. Постановка задачи 2](#_Toc87651434)

[2. Описание используемых функций 2](#_Toc87651435)

[3. Реализация 2](#_Toc87651436)

[Заключение 9](#_Toc87651437)

# 1. Постановка задачи

Посмотрите в прилагаемом материале еще раз примеры сборки распределенных по процессам данных при помощи коллективных функций MPI\_Gather, MPI\_Allgather, MPI\_Gatherv, MPI\_Allgatherv. Выполните упражнения, приведенные в конце параграфа - 6 задач. Оформляете в одном отчете - номер задачи, текст условия, текст программы с поясняющими комментариями в программе, скрин компиляции и скрин запуска на несколько процессов. Удачи!

# 2. Описание используемых функций

MPI\_Gather – позволяет собирать данные одинаковой длины с разных процессов на один процесс.

MPI\_Gatherv – позволяет собирать данные разной длины с разных процессов на один процесс.

# 3. Реализация

Задание 1.

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.

Программный код:

#include <stdio.h>

#include "mpi.h"

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

    int rank;

    int size;

    int i,s=0;

    MPI\_Init(&argc, &argv);

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

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

    int \*b=new int [3];

    for(i=0;i<3;i++)

    b[i]=rank;

    int \*a=new int[3\*size];

    MPI\_Gather(b,3,MPI\_INT,a,3,MPI\_INT,0,MPI\_COMM\_WORLD);

    for(i=0; i<100000000\*rank; i++)

        s+=1;

    printf("rank= %d  a: ",rank);

    for(i=0; i<3\*size; i++)

    printf(" %d ",a[i]);

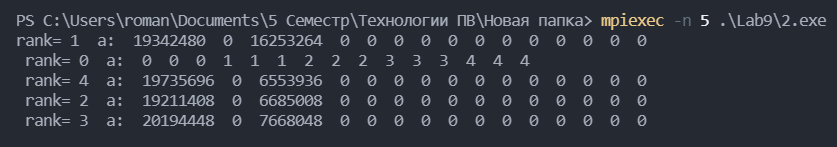
    printf("\n ");

    MPI\_Finalize();

    return 0;

}

Выполнение:



Задание 2.

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).

Программный код:

#include <stdio.h>

#include "mpi.h"

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

{

int rank;

int size;

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

b[i]=rank;

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

int \*dl = new int[size];

int \*ds = new int [size];

int n = 0;

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

{

    dl[i]=size - i;

    ds[i]=n;

    n += RC[i];

}

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

for(i=0; i<100000000\*rank; i++)

s+=1;

printf("rank= %d  a: ",rank);

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

printf("  %d  ",a[i]);

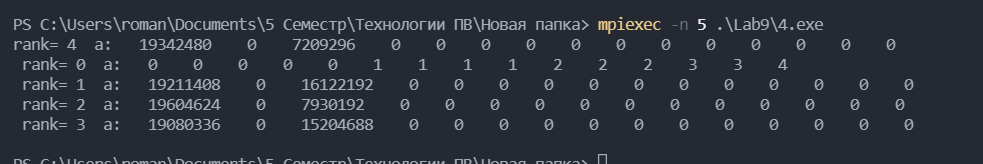
printf("\n ");

MPI\_Finalize();

return 0;

}

Выполнение:



Задание 3.

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.

Программный код:

#include <stdio.h>

#include "mpi.h"

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

int rank;

int size;

int n, s = 0;

MPI\_Init(&argc, &argv);

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

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

if (rank == 0)

scanf("%d", &n);

MPI\_Bcast(&n, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

int\* b = new int[n\*2];

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

    b[i] = rank;

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

for (int i = 1; i < size; i++)

    a[i] = a[i - 1] + n;

MPI\_Gather(b, 2\*n, MPI\_INT, a, 2\*n, MPI\_INT, 0, MPI\_COMM\_WORLD);

if (rank == 0) {

    printf("rank = %d a:\n", rank);

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

        for (int j = 0; j < n; j++)

            printf("%d ",a[j+i\*n]);

        printf("\n");

    }

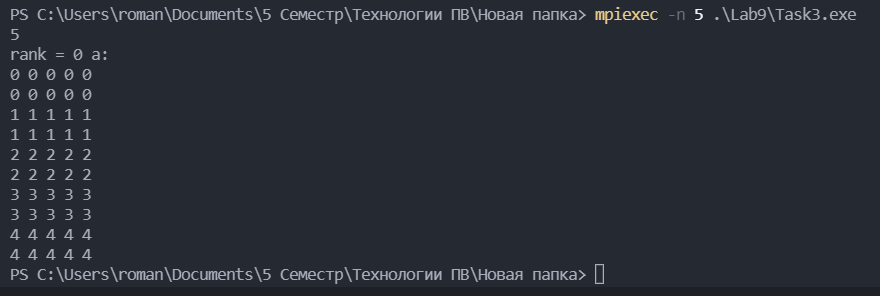
}

MPI\_Finalize();

return 0;

}

Выполнение:



Задание 4.

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

Программный код:

#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[size];

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

    b[i] = rank;

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

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

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

    a[i] = a[i - 1] + size;

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

    for (j = 0; j < size; j++) {

        a[i][j] = -1;

    }

}

int\* ds = new int[size];

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

    ds[i] = (i \* size) + i;

for (int l = 0; l < 2; l++) {

    int\* dl = new int[size];

    int\* ds = new int[size];

    for (int i = 0; i < size; i++)

    {

        dl[i] = i + 1;

        ds[i] = (i \* size \* 2) + l \* size;

    }

    MPI\_Gatherv(b, rank + 1, MPI\_INT, \*a, dl, ds, MPI\_INT, 0, MPI\_COMM\_WORLD);

}

    if (rank == 0){

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

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

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

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

            printf("\n");

        }

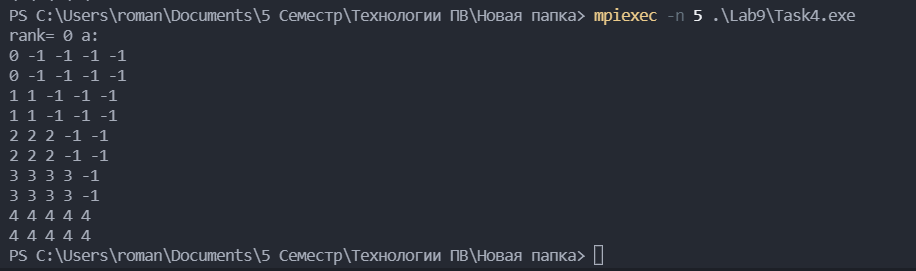
    }

    MPI\_Finalize();

    return 0;

}

Выполнение:



Задание 5.

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.

Программный код:

#include <stdio.h>

#include "mpi.h"

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

    int rank;

    int size;

    int s = 0;

    MPI\_Init(&argc, &argv);

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

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

    int\* dl = new int[size];

    int\* ds = new int[size];

    int\* b = new int[size];

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

        b[i] = rank;

    int\*\* a = new int\* [size];

    a[0] = new int[size \* size];

    for (int i = 1; i < size; i++)

        a[i] = a[i - 1] + size;

    for (int i = 0; i < size; i++) {

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

        {

            a[i][j] = 0;

        }

    }

    for (int i = 0; i < size; i++) {

        dl[i] = size - i;

        ds[i] = i \* size;

    }

    MPI\_Gatherv(b, size - rank, MPI\_INT, \*a, dl, 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++)

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

            printf("\n");

        }

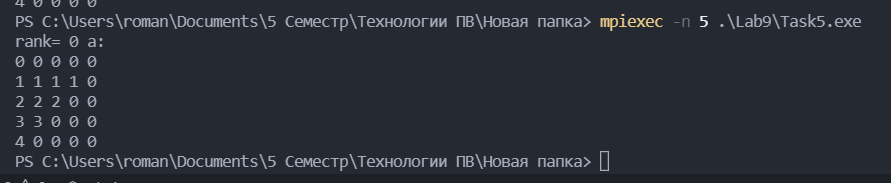
    }

    MPI\_Finalize();

    return 0;

}

Выполнение:



Задание 6.

Модифицируйте предыдущий пример, осуществляя сохранение каждой   
строки, начиная с элемента на главной диагонали.

Программный код:

#include <stdio.h>

#include "mpi.h"

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

{

    int rank;

    int size;

    int s = 0;

    MPI\_Init(&argc, &argv);

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

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

    int\* b = new int[size];

    int di = size - (size - rank);

    int\*\* mas = new int\* [size];

    int\* mas1 = new int[size];

    int\* mas2 = new int[size];

    for (int i = 0; i < di; i++)

        b[i] = 0;

    for (int i = di; i < size; i++)

        b[i] = rank;

    mas[0] = new int[size \* size];

    for (int i = 1; i < size; i++)

        mas[i] = mas[i - 1] + size;

    for (int i = 0; i < size; i++)

    {

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

        {

            mas[i][j] = 0;

        }

    }

    for (int i = 0; i < size; i++) {

        mas1[i] = size;

        mas2[i] = i \* size;

    }

    MPI\_Gatherv(b, size, MPI\_INT, \*mas, mas1, mas2, MPI\_INT, 0, MPI\_COMM\_WORLD);

    if (rank == 0)

    {

        printf("rank = %d\n", rank);

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

        {

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

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

            printf("\n");

        }

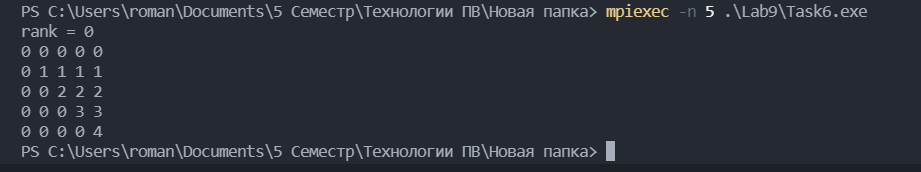
    }

    MPI\_Finalize();

    return 0;

}

Выполнение:



# Заключение

В результате выполнения работы изучил новые функции с помощью которых можно собирать информацию с разных процессов на один.