МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

**Институт фундаментальных наук**

**ДОМАШНЯЯ РАБОТА №8**

**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](#_Toc87042346)

[2. Описание используемых функций 2](#_Toc87042347)

[3. Реализация 2](#_Toc87042348)

[Заключение 5](#_Toc87042349)

# 1. Постановка задачи

Создайте собственную функцию широковещательной рассылки (аналог MPI\_Bcast), но построенную с использованием блокирующих функций парного обмена сообщениями (MPI\_Send, MPI\_Recv). Рассылку сообщения реализовать по 2-му дереву: сначала осуществляется одна пересылка (например, 0-й процесс отсылает сообщение 1-му), затем - две пересылки (например, 0-й и 1-й осуществляют отправки сообщений 2-му и 3-му), и т.д. Написать универсальную функцию (для любого количества процессов, для любого источника рассылки, для любых типов данных и любого количества пересылаемых элементов, на каждый указанный случай привести результат тестирования). В момент запуска параллельного приложения соберите трассу выполнения программы и приведите рисунок трассы в отчете (кол-во процессов небольшое, но достаточное для наглядного подтверждения корректности способа рассылки). Проведите сравнение времени выполнения созданной Вами функции с функцией MPI\_Bcast, варьируя количество рассылаемых элементов и количество задействованных процессов.

# 2. Описание используемых функций

MPI\_Bcast - Функция широковещательной передачи.

# 3. Реализация

Реализация с использованием собственной функции:

#include <stdio.h>

#include "mpi.h"

void bcast(void\*, int, MPI\_Datatype, int, int, int);

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

{

    int rank;

    int size;

    int buf;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    if (rank == 5)

    {

        buf = 3;

    }

    bcast(&buf, 1, MPI\_INT, 5, rank, size);

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

    MPI\_Finalize();

    return 0;

}

void bcast(void\* buf, int n, MPI\_Datatype type, int root, int rk, int sz)

{

    MPI\_Status stat1;

    int R;

    if (rk == root)

    {

        for (int i = 0; i < log(sz) / log(2); i++)

        {

            R = pow(2, i) + rk;

            if (R > sz - 1)

                R = R - sz - 1;

            MPI\_Send(buf, n, type, R, 777, MPI\_COMM\_WORLD);

        }

    }

    else

    {

        MPI\_Recv(buf, n, type, MPI\_ANY\_SOURCE, 777, MPI\_COMM\_WORLD, &stat1);

        for (int i = 0; i < log(sz) / log(2); i++)

        {

            R = pow(2, i) + rk;

            if (R > sz - 1)

                R = R - sz - 1;

            MPI\_Send(buf, n, type, R, 777, MPI\_COMM\_WORLD);

        }

    }

}

Реализация с использованием штатной функции:

#include <stdio.h>

#include "mpi.h"

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

{

    int rank;

    int size;

    int buf;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    if (rank == 5)

    {

        buf = 3;

    }

    MPI\_Bcast(&buf, 1, MPI\_INT, 5, MPI\_COMM\_WORLD);

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

    MPI\_Finalize();

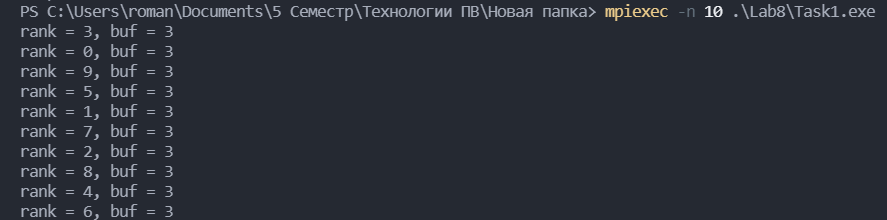
    return 0;

}

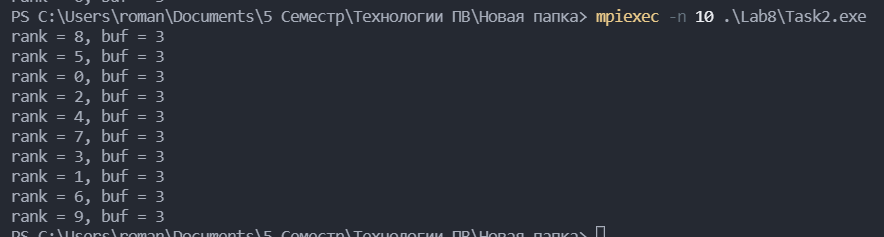
Пример запуска:

Собственная функция:

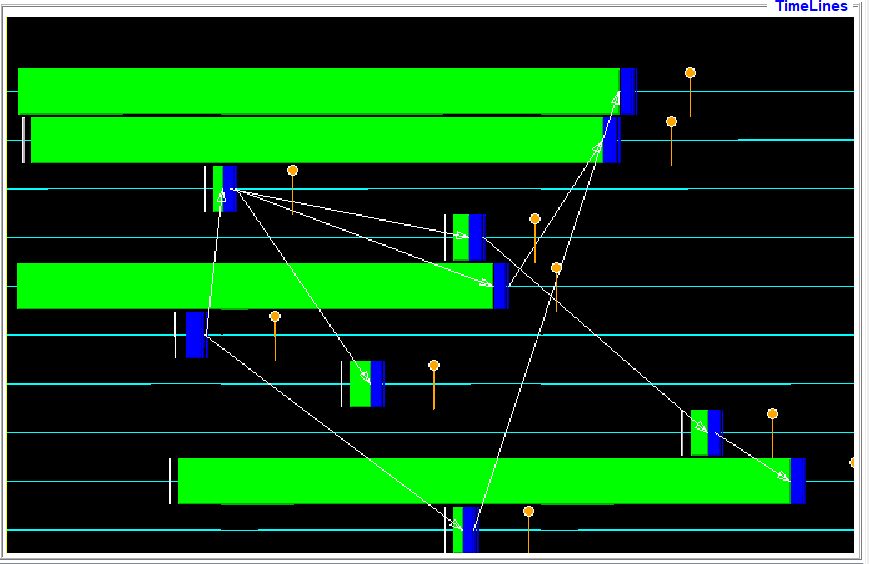
Рассылка идет с 5-го процесса:



Штатная функция:

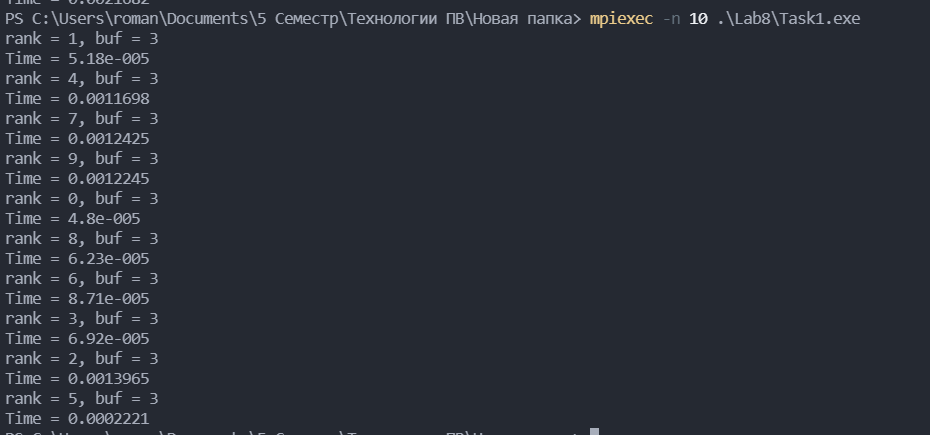


Трасса:

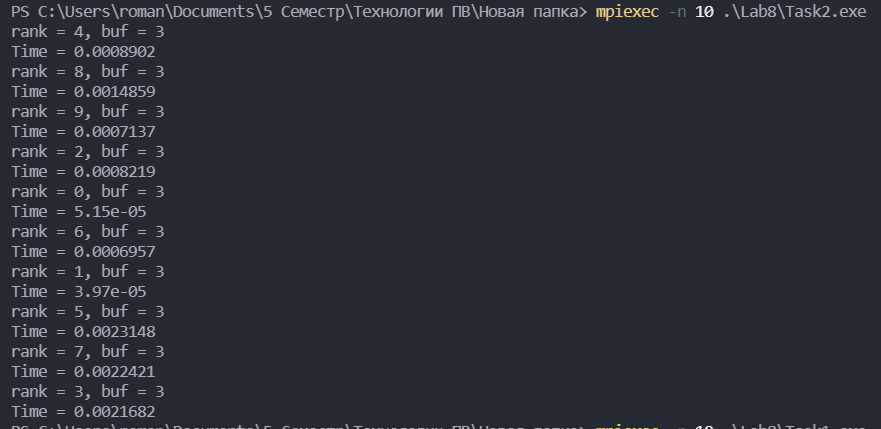


Сравнение времени выполнения:

Функция собственного написания:



Штатная функция:



# Заключение

Научились использовать функцию широковещательной передачи данных.  
Так можно сделать вывод, что наша функция получилась быстрее. Я думаю, что это получилось из-за того, что в штатной функции предусмотрено большое количество на валидность значений, чего нет в нашей функции.