МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

**Институт фундаментальных наук**

**ДОМАШНЯЯ РАБОТА №12**

**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](#_Toc89455064)

[2. Описание используемых функций 2](#_Toc89455065)

[3. Реализация 2](#_Toc89455066)

[Заключение 7](#_Toc89455067)

# 1. Постановка задачи

Требуется провести исследование по способам передачи данных четных элементов одномерного массива. Для этого напишите несколько программ передачи данных с 0-го процесса на 1-й и обратно с замером времени на 0-м процессе перед отправкой и после получения. Размер передаваемого массива должен быть значительным (10^3,^4,^5...^9), чтобы можно было пренебречь погрешностью измерения времени. 1-я программа: массив передается целиком одной посылкой. 2-я программа: передаются в цикле по одному элементу только четные элементы. 3-я программа: перед отправкой четные элементы переприсваиваются в промежуточный массив половинной длины, он передается и после получения снова переприсваивается в исходный массив. 4-я программа: при помощи конструктора MPI\_Type\_vector создается новый описатель расположения данных в ОП, передача осуществляется с его помощью. Результат измерения времени при разных вариантах передачи данных приведите в таблице. Укажите плюсы и минусы каждого из рассмотренных подходов.

# 2. Описание используемых функций

MPI\_Type\_vector – создает произвольный тип данных, где блоки данных разнесены друг от друга на одинаковое расстояние.

# 3. Реализация

1. Массив передается целиком одной посылкой.

Программный код:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "mpi.h"

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

    int rank, size;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    double t1,t2;

    int n,\*a;

    for (int i=3;i<10;i++) {

        n = pow(10,i);

        a = new int[n];

        if (rank==0) {

            for (int i=0;i<n;i++) {

                a[i]=rand();

        }

        t1=MPI\_Wtime();

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

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

        t2=MPI\_Wtime();

        printf("Size: 10^%d Time: %lf\n", i, (t2-t1)/2);

        }

    else {

        MPI\_Recv(a, n, MPI\_INT, 0, 777, MPI\_COMM\_WORLD, &stat);

        MPI\_Send(a, n, MPI\_INT, 0, 777, MPI\_COMM\_WORLD);

    }

    delete a;

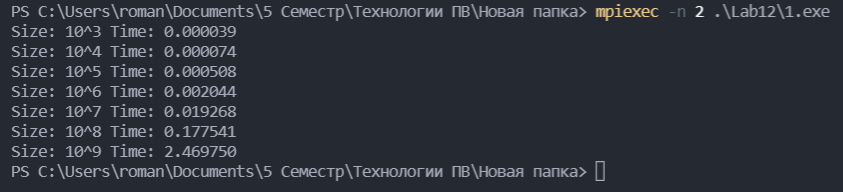
    }

    MPI\_Finalize();

    return 0;

}

Выполнение:



2. В цикле по одному элементу:

Программный код:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "mpi.h"

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

    int rank, size;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    double t1,t2;

    int n,\*s;

    for (int i=3;i<10;i++) {

        n=pow(10,i);

        s=new int[n];

        if (rank==0) {

            for (int i=0;i<n;i++) {

                s[i]=rank;

            }

            t1=MPI\_Wtime();

            for (int i=0;i<n;i+=2) {

                MPI\_Send(&s[i],1,MPI\_INT,1,777,MPI\_COMM\_WORLD);

            }

            for (int i=0;i<n;i+=2) {

                MPI\_Recv(&s[i],1,MPI\_INT,1,777,MPI\_COMM\_WORLD,&stat);

            } t2=MPI\_Wtime();

            printf("Size: 10^%d Time: %lf\n", i, (t2-t1)/2);

        } else {

            for (int i=0;i<n;i+=2)

                MPI\_Recv(&s[i],1, MPI\_INT,0,777,MPI\_COMM\_WORLD,&stat);

            for (int i=0;i<n;i+=2)

                MPI\_Send(&s[i],1, MPI\_INT,0,777,MPI\_COMM\_WORLD);

            }

        delete s;

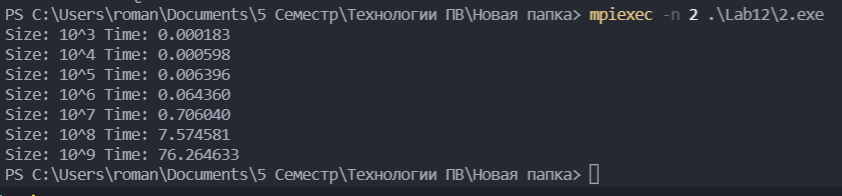
    }

    MPI\_Finalize();

    return 0;

}

Выполнение:



3. Переприсваивание

Программный код:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "mpi.h"

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

{

    int rank, size;

    MPI\_Status stat;

    MPI\_Init(&argc, &argv);

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

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

    double t1, t2;

    int n, \*s, \*s1;

    for (int i = 3; i < 10; i++)

    {

        n = pow(10, i);

        s = new int[n];

        s1 = new int[n / 2];

        if (rank == 0)

        {

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

            {

                s[i] = rank;

            }

            t1 = MPI\_Wtime();

            for (int i = 0, x = 0; i < n; i += 2, x++)

            {

                s1[x] = s[i];

            }

            MPI\_Send(s1, n / 2, MPI\_INT, 1, 777, MPI\_COMM\_WORLD);

            MPI\_Recv(s1, n / 2, MPI\_INT, 1, 777, MPI\_COMM\_WORLD, &stat);

            t2 = MPI\_Wtime();

            printf("Size: 10^%d Time: %lf", i, (t2 - t1) / 2);

        }

        else

        {

            MPI\_Recv(s1, n / 2, MPI\_INT, 0, 777, MPI\_COMM\_WORLD, &stat);

            for (int i = 0, x = 0; i < n; i += 2, x++)

            {

                s[i] = s1[x];

            }

            MPI\_Send(s1, n / 2, MPI\_INT, 0, 777, MPI\_COMM\_WORLD);

        }

        delete s;

        delete s1;

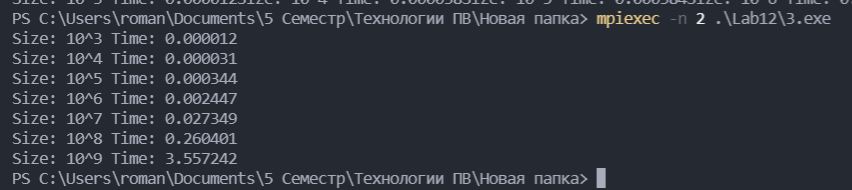
    }

    MPI\_Finalize();

    return 0;

}

Выполнение:



4. MPI\_vtype

Программный код:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "mpi.h"

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

{

    int rank, size;

    MPI\_Status status;

    MPI\_Init(&argc, &argv);

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

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

    double t1, t2;

    int n, \*s;

    MPI\_Datatype MPI\_Mytype;

    for (int i = 3; i < 10; i++)

    {

        n = pow(10, i);

        s = new int[n];

        MPI\_Type\_vector(n / 2, 1, 2, MPI\_INT, &MPI\_Mytype);

        MPI\_Type\_commit(&MPI\_Mytype);

        if (rank == 0)

        {

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

            {

                s[i] = rank;

            }

            t1 = MPI\_Wtime();

            MPI\_Send(s, 1, MPI\_Mytype, 1, 777, MPI\_COMM\_WORLD);

            MPI\_Recv(s, 1, MPI\_Mytype, 1, 777, MPI\_COMM\_WORLD, &status);

            t2 = MPI\_Wtime();

            printf("Size: 10^%d Time: %lf\n", i, (t2 - t1) / 2);

        }

        else

        {

            MPI\_Recv(s, 1, MPI\_Mytype, 0, 777, MPI\_COMM\_WORLD, &status);

            MPI\_Send(s, 1, MPI\_Mytype, 0, 777, MPI\_COMM\_WORLD);

        }

        t1 = 0;

        t2 = 0;

        delete s;

    }

    MPI\_Finalize();

    return 0;

}

Выполнение:

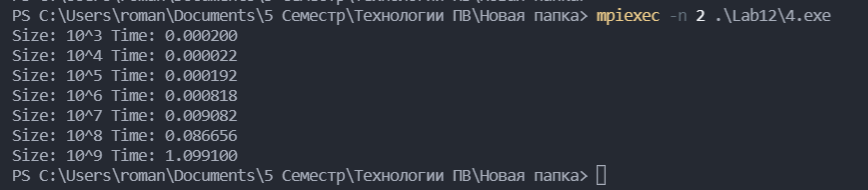


Таблица результатов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер ряда: | 1 программа | 2 программа | 3 программа | 4 программа |
| 10^3 | 0.000039 | 0.000183 | 0.000012 | 0.000200 |
| 10^4 | 0.000074 | 0.000598 | 0.000031 | 0.000022 |
| 10^5 | 0.000508 | 0.006396 | 0.000344 | 0.000192 |
| 10^6 | 0.002044 | 0.064360 | 0.002447 | 0.000818 |
| 10^7 | 0.019268 | 0.706040 | 0.027349 | 0.009082 |
| 10^8 | 0.177541 | 7.574581 | 0.260401 | 0.086656 |
| 10^9 | 2.469750 | 76.264633 | 3.557242 | 1.099100 |

# Заключение

Выполнив лабораторную работу и проанализировав результаты можно сделать выводы. Самый медленный способ это по элементная передача массива. Средний по скорости это передача всего массива целиком. Самый быстрый это через Type. При помощи карт можно на много быстрее передать данные.