Лабораторная работа № 3 Основы MPI	ФИО	Титов А.К.
	Группа	ИВТ 460
	Предмет	Параллельное программирование
	Дата отчета	
	Оценка	
	Подпись преподавателя	

Постановка задачи

1. Напишите простейшую MPI- программу, которая выводит номера и общее число запущенных процессов. Запустите ее с разным числом процессов.

2. Напишите программу, инициализирующую многопоточную среду MPI. Запустите ее с разным числом процессов, создавая различное число потоков. Попробуйте использовать различные уровни поддержки многопоточности.

3. Исследование блокирующих операций обмена типа точка – точка. Напишите программу, которая передает значение, сгенерированное в процессе 0, по кругу через все потоки обратно в процесс 0. Измерьте время, потраченное на передачу. Запустите программу с различным числом процессов. Измените объем передаваемых данных.

4. Напишите программу, демонстрирующую отличие между стандартной отправкой сообщение MPI_Send и отправкой с синхронизацией MPI_Ssend.

5. Изменить программу из п.3 для использования буферизованной отправки сообщений MPI_Bsend.

6. Напишите программу, иллюстрирующую применение функций MPI_Probe, MPI_Get_count.

7. Изменить программу из п.3 для использования неблокирующих функций отправки и приема сообщений, а также функций MPI_Wait, MPI_Test .

Результаты

Для 2 ядер

```
D:\My\Workspace\C++\MPI Multiprocessing\Debug>mpiexec.exe -n 2 "MPI Multiprocessing.exe"
Task1
====================
1
Size is 2
0
Time: 0.000400693

Task3
====================
Proc 1 got 15 from 0
Got 15
Time: 0.000122731

Task4
====================
Send time is 8.98032e-06
Ssend time is 4.27634e-05
Time: 0.000151383

Task5
====================
Proc 1 got from 0
0 0 0 0 0 0 0 0 0 0
Time: 0.000259574

Task6
====================
Proc 1 got from 0 message with count 30
I  l o v e  c h e e s e .  I  n e e d  c h e e s e !
Proc 1 waiting for recieve from 0
Time: 1.24014e-05

Task7
====================
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Time: 0.108464

Individual Task
====================
Function 1 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184
0.893997 -0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Time: 0.000407536
```

Для 4 ядер

```
D:\My\Workspace\C++\MPI Multiprocessing\Debug>mpiexec.exe -n 4 "MPI Multiprocessing.exe"
Task1
1
3
2
====================
Size is 4
0
Time: 0.000403687

Task3
====================
Proc 1 got 15 from 0
Proc 2 got 15 from 1
Proc 3 got 15 from 2
Got 15
Time: 0.000265561

Task4
====================
Send time is 0.000230923
Ssend time is 0.000181745

Task5
====================
Proc 1 got from 0
Proc 2 got from 1
Proc 3 got from 2
0 0 0 0 0 0 0 0 0 0
Time: 0.000465266

Task6
====================
Proc 1 got from 0 message with count 30
I  l o v e  c h e e s e .  I  n e e d  c h e e s e !
Proc 2 waiting for recieve from 1
Proc 3 waiting for recieve from 2
Time: 6.84215e-06
Proc 1 waiting for recieve from 0

Task7
====================
Proc 3 waiting for recieve from 2
Proc 2 waiting for recieve from 1
Proc 3 waiting for recieve from 2
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Time: 0.352192

IndividualTask
====================
Function 2 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 3 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 1 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Time: 0.00192649
```

Для 8 ядер

```
D:\My\Workspace\C++\MPI Multiprocessing\Debug>mpiexec.exe -n 8 "MPI Multiprocessing.exe"
7
2
4
6
5
3
1
Task1
====================
Size is 8
0
Time: 0.000691057

Task3
====================
Proc 1 got 15 from 0
Proc 2 got 15 from 1
Proc 3 got 15 from 2
Proc 4 got 15 from 3
Proc 5 got 15 from 4
Proc 6 got 15 from 5
Proc 7 got 15 from 6
Got 15
Time: 0.000712011

Task4
====================
Send time is 0.000223653
Ssend time is 0.000210824


Task5
====================
Proc 1 got from 0
Proc 2 got from 1
Proc 3 got from 2
Proc 4 got from 3
Proc 5 got from 4
Proc 6 got from 5
Proc 7 got from 6
0 0 0 0 0 0 0 0 0 0
Time: 0.000950204

Task6
====================
Proc 1 got from 0 message with count 30
I  l o v e  c h e e s e .  I  n e e d  c h e e s e !
Proc 1 waiting for recieve from 0
Proc 2 waiting for recieve from 1
Proc 3 waiting for recieve from 2
Proc 4 waiting for recieve from 3
Proc 5 waiting for recieve from 4
Proc 6 waiting for recieve from 5
Proc 7 waiting for recieve from 6
Time: 8.55269e-06

Task7
====================
Proc 4 waiting for recieve from 3
Proc 7 waiting for recieve from 6
Proc 5 waiting for recieve from 4
Proc 6 waiting for recieve from 5
Proc 2 waiting for recieve from 1
Proc 3 waiting for recieve from 2
Proc 6 waiting for recieve from 5
Proc 5 waiting for recieve from 4
Proc 4 waiting for recieve from 3
Proc 7 waiting for recieve from 6
Proc 6 waiting for recieve from 5
Proc 5 waiting for recieve from 4
Proc 7 waiting for recieve from 6
Proc 6 waiting for recieve from 5
Proc 7 waiting for recieve from 6
Proc 7 waiting for recieve from 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Time: 0.692544

IndividualTask
====================
Function 4 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 2 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 6 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 7 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 1 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 3 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 5 calculated
0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -
0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Time: 0.00469414
```

Код программы

Тестирующая функция

```
#include "mpi.h"
#include <iostream>

using namespace std;
void Task1();
void Task3();
void Task4();
void Task5();
void Task6();
void Task7();
void IndividualTask();

int main(int argc, char * argv[])
{
    double start, end;


    MPI_Init(&argc, &argv);
    start = MPI_Wtime();
    IndividualTask();
    end = MPI_Wtime();

    // Print time (master)
    int procRank;
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);
    if (procRank == 0)
        cout << "Time: " << end - start;

    MPI_Finalize();
}
```

Задание 1, 2

```
// Print proc size
void Task1()
{
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    if (procRank == 0)
        cout << "Size is " << procSize << endl;

    cout << procRank << endl;
}
```

Задание 3

```
// Send number in a circle
void Task3()
{
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    bool isMain = procRank == 0;
    if (isMain)
    {
        int buffer = 15;
        int recieverRank = (procRank + 1) % procSize;
        MPI_Send(&buffer, 1, MPI_INT, recieverRank, 0, MPI_COMM_WORLD);
        int lastRank = procSize - 1;

        // Other procs work
        MPI_Status recvStatus;
        int gotFromLast;
        MPI_Recv(&gotFromLast, 1, MPI_INT, lastRank, MPI_ANY_TAG, MPI_COMM_WORLD, &recvStatus);
        cout << "Got " << gotFromLast << endl;
    }
    else
    {
        int buffer;
        MPI_Status recvStatus;
        int senderRank = procRank - 1;
        MPI_Recv(&buffer, 1, MPI_INT, senderRank, MPI_ANY_TAG, MPI_COMM_WORLD, &recvStatus);
        cout << "Proc " << procRank << " got " << buffer << " from " << senderRank << endl;

        int recieverRank = (procRank + 1) % procSize;
        MPI_Send(&buffer, 1, MPI_INT, recieverRank, 0, MPI_COMM_WORLD);
    }
}


void SendTest() {
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    // send big data and watch for time of execution (in 2 different programms)
    double start = MPI_Wtime();
    bool isMain = procRank == 0;
    if (isMain)
    {
        int data[1000] = { 0 };
        MPI_Send(data, 1000, MPI_INT, 1, 0, MPI_COMM_WORLD);
    }
    else {
        MPI_Status status;
        int data[1000];
        MPI_Recv(data, 1000, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &status);
    }
    double finish = MPI_Wtime();

    if (isMain)
        cout << "Send time is " << finish - start << endl;
}

void SsendTest()
{
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    double start = MPI_Wtime();
    bool isMain = procRank == 0;
    if (isMain)
    {
        int data[1000] = { 0 };
        MPI_Ssend(data, 1000, MPI_INT, 1, 0, MPI_COMM_WORLD);
    }
    else {
        MPI_Status status;
        int data[1000];
        //MPI_Recv(data, 1000, MPI_INT, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &status); если эту строку закомментировать, то программа
зависнет
    }
    double finish = MPI_Wtime();

    if (isMain)
        cout << "Ssend time is " << finish - start << endl;
}
```

Задание 4

```
// Sync send big data
// Or thread not recieve sync send package
void Task4()
{
    SendTest();
    SsendTest();
}


template <class T>
void PrintArray(T * arr, int size)
{
    for (int i = 0l; i < size; i++)
        cout << arr[i] << ' ';

    cout << endl;
}
```

Задание 5

```
// BSend with buffer attachment
// Recv array, send array
void Task5()
{
    const int dataSize = 10;
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    bool isMain = procRank == 0;
    if (isMain)
    {
        int data[dataSize] = { 0 };
        int buffSize = dataSize * sizeof(int) + MPI_BSEND_OVERHEAD;
        char * buffer = (char *) malloc(buffSize);
        MPI_Buffer_attach(buffer, buffSize);

        int recieverRank = (procRank + 1) % procSize;
        MPI_Bsend(&data, dataSize, MPI_INT, recieverRank, 0, MPI_COMM_WORLD);
        int lastRank = procSize - 1;

        // Other procs work

        MPI_Status recvStatus;
        MPI_Recv(&data, dataSize, MPI_INT, lastRank, MPI_ANY_TAG, MPI_COMM_WORLD, &recvStatus);

        MPI_Buffer_detach(buffer, &buffSize);
        free(buffer);

        PrintArray(data, dataSize);
    }
    else
    {
        int data[dataSize];

        MPI_Status recvStatus;
        int senderRank = procRank - 1;
        MPI_Recv(&data, dataSize, MPI_INT, senderRank, MPI_ANY_TAG, MPI_COMM_WORLD, &recvStatus);
        cout << "Proc " << procRank << " got from " << senderRank << endl;

        int recieverRank = (procRank + 1) % procSize;

        int buffSize = dataSize * sizeof(int) + MPI_BSEND_OVERHEAD;
        char * buffer = (char *)malloc(buffSize);
        MPI_Buffer_attach(buffer, buffSize);

        MPI_Bsend(&data, dataSize, MPI_INT, recieverRank, 0, MPI_COMM_WORLD);

        MPI_Buffer_detach(buffer, &buffSize);
        free(buffer);
    }
}
```

Задание 6

```
// Send to all
// Reciecieve from anyone withour knowledge
// about count of elements or source
void Task6()
{
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    bool isMain = procRank == 0;
    if (isMain)
    {
        char data[100] = "I love cheese. I need cheese!";
        MPI_Send(data, strlen(data) + 1, MPI_CHAR, 1, 0, MPI_COMM_WORLD);
    }
    else if (procRank == 1)
    {
        MPI_Status recvStatus;
        MPI_Probe(MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &recvStatus);
        int source = recvStatus.MPI_SOURCE;
        int count = recvStatus.count;

        char *data = new char[count];
        MPI_Recv(data, count, MPI_CHAR, source, 0, MPI_COMM_WORLD, &recvStatus);
        cout << "Proc " << procRank << " got from " << source << " message with count " << count <<
endl;
        PrintArray<char>(data, count);
        delete[] data;
    }
}
```

Задание 7

```
#include <Windows.h>
// http://rsusu1.rnd.runnet.ru/tutor/method/m2/page07.html
// Reciever waits and performs calculations
// And writes message
void Task7()
{
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    bool isMain = procRank == 0;
    if (isMain)
    {
        const int bufferSize = 100;
        int buffer[bufferSize] = {1, 2, 3, 4, 5, 6};

        // Send
        int recieverRank = (procRank + 1) % procSize;
        MPI_Request sendRequest;
        MPI_Status sendStatus;
        MPI_Isend(&buffer, bufferSize, MPI_INT, recieverRank, 0, MPI_COMM_WORLD, &sendRequest);
        MPI_Wait(&sendRequest, &sendStatus);

        // Recv
        MPI_Status recvStatus;
        MPI_Request recvRequest;
        int lastRank = procSize - 1;
        int inBuffer[bufferSize] = {};

        MPI_Irecv(&buffer, bufferSize, MPI_INT, lastRank, MPI_ANY_TAG, MPI_COMM_WORLD,
&recvRequest);
        MPI_Wait(&recvRequest, &recvStatus);

        PrintArray(inBuffer, bufferSize);
    }
    else
    {
        // Recv
        const int bufferSize = 100;
        int buffer[bufferSize];

        MPI_Request recvRequest;
        int senderRank = procRank - 1;
        MPI_Irecv(&buffer, bufferSize, MPI_INT, senderRank, MPI_ANY_TAG, MPI_COMM_WORLD,
&recvRequest);

        MPI_Status recvStatus;
        int recvFinished;
        do {
            cout << "Proc " << procRank << " waiting for recieve from " << senderRank << endl;
            // While recv is performing, we can do other things here
            Sleep(100); // Calculations
            MPI_Test(&recvRequest, &recvFinished, &recvStatus);
        } while (!recvFinished);


        // Send
        int recieverRank = (procRank + 1) % procSize;
        MPI_Request sendRequest;
        MPI_Status sendStatus;
        MPI_Isend(&buffer, bufferSize, MPI_INT, recieverRank, 0, MPI_COMM_WORLD, &sendRequest);
        MPI_Wait(&sendRequest, &sendStatus);
    }
}
```

Индивидуальное задание

```
#include <vector>
using std::vector;
typedef double(*matrFunc)(double); // math func
void IndividualTask()
{
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    const int rangeLength = 20;
    vector<matrFunc> functions(procSize, sin); // many sinuses

    bool isMain = procRank == 0;
    if (isMain)
    {
        // Create Range
        double range[rangeLength] = {};
        double start = 0.0;
        double finish = 150.0;
        double step = abs(start - finish) / rangeLength;
        for (int i = 0; i < rangeLength; i++)
            range[i] = start + i*step;

        MPI_Bcast(range, rangeLength, MPI_DOUBLE, 0, MPI_COMM_WORLD);

        double * functionValues = new double[rangeLength];
        for (int i = 1; i < procSize; i++)
        {
            MPI_Status recvStatus;
            MPI_Recv(functionValues, rangeLength, MPI_DOUBLE, MPI_ANY_SOURCE, MPI_ANY_TAG,
MPI_COMM_WORLD, &recvStatus);
            cout << "Function " << recvStatus.MPI_SOURCE << " calculated" << endl;
            PrintArray(functionValues, rangeLength);
        }
    }
    else
    {
        double * range = new double[rangeLength];
        MPI_Bcast(range, rangeLength, MPI_DOUBLE, 0, MPI_COMM_WORLD);


        double *functionValues = new double[rangeLength];

        //Calculations
        for (int i = 0; i < rangeLength; i++)
            functionValues[i] = functions[procRank](range[i]);

        MPI_Send(functionValues, rangeLength, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);

        delete[] range;
        delete[] functionValues;
    }
}
```