МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЁТ

по лабораторной работе №1-2

по дисциплине «Параллельные алгоритмы и системы»

Тема: Запуск параллельной программы и передача данных по процессам

Студенты гр. 1307 _____ Таланков В.Р.

Преподаватель _____

Санкт-Петербург

2025

Цель работы

Освоить процесс запуска программы на Java с применением библиотеки MPJ. Научиться получать сведения о количестве запущенных процессов и номере отдельного процесса.

Освоить функции передачи данных между процессами.

Лабораторная работа №1

Задание на лабораторную работу

Задание 1.

Создать и запустить программу на 2-х процессах с применением функций int MPI_Init.

Задание 2.

Создать и запустить программу на 3-х процессах с применением функций:

1) int MPI_Init(int* argc, char*** argv);

2) int MPI_Finalize(void);

3) int MPI_Comm_size(MPI_Comm comm, int* size)

4) int MPI_Comm_rank(MPI_Comm comm, int* rank)

Программа должна выводить на экран номер процесса и како1-либо идентификатор процесса.

Задание 3.

Создать и запустить программу на n-х процессах печати таблицы умножения.

Ход работы

В ходе выполнения работы был реализован итоговый файл, покрывающий все задания разом. Запуск данного файла производится через командную строку внутри Intellij IDEA.

Процесс вычисляет таблицу умножения, где множитель может быть числом от 2 до 10 включительно. Таким образом мы имеем 9 чисел и 81 результат.

```
PS C:\Users\npc\Desktop\parallel-labs> java -jar ".\mpj-v0_44\lib\starter.jar" -np 4 -cp ".\target\classes" neko.lab1.part1
MPJ Express (0.44) is started in the multicore configuration
P0 (id=21): 2*2=4
P1 (id=23): 3*2=6
P2 (id=24): 4*2=8
P3 (id=22): 5*2=10
P0 (id=21): 2*3=6
P1 (id=23): 3*3=9
P2 (id=24): 4*3=12
P3 (id=22): 5*3=15
P0 (id=21): 2*4=8
P1 (id=23): 3*4=12
P2 (id=24): 4*4=16
P3 (id=22): 5*4=20
P0 (id=21): 2*5=10
P1 (id=23): 3*5=15
P2 (id=24): 4*5=20
P3 (id=22): 5*5=25
```

Рисунок 1 – Результат работы программы (4 процесса)

Код программы (lab1.part1.java)

```
package neko.lab1;

import mpi.MPI;

public class part1 {
    public static void main(String[] args) {
        MPI.Init(args);

        int rank = MPI.COMM_WORLD.Rank();
        int size = MPI.COMM_WORLD.Size();

        for (int i = rank + 2; i <= 10; i += size) {
            for (int j = 2; j <= 10; j++) {
                System.out.println("P" + rank + " (id=" +
Thread.currentThread().getId()
                        + "): " + i + "*" + j + "=" + (i * j));
            }
        }

        MPI.Finalize();
    }
}
```

Лабораторная работа №2

Задание на лабораторную работу

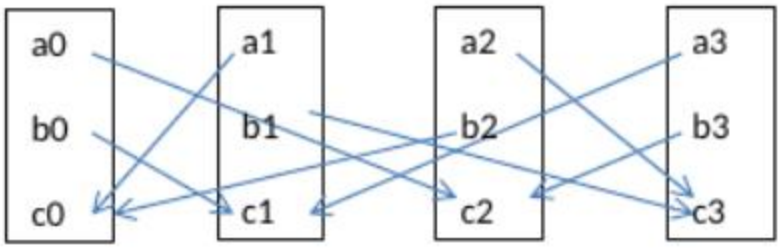
Задание 1.

1) Запустить 4 процесса.

2) На каждом процессе создать переменные: ai,bi,ci, где I – номер процесса.

Инициализировать переменные. Вывести данные на печать.

3) Передать данные на другой процесс. Напечатать номера процессов и поступившие данные. Найти: c0=a1+b2; c1=a3+b0; c2=a0+b3; c3=a2+b1.



Задание 2.

Запустить n процессов и найти по вариантам: Сумму нечетных элементов вектора;

Ход работы

Были созданы два отдельных приложения. Приложения для задания 2 требует обязательного указания параметра –Dvector (имеется значения по умолчанию).

Первое приложение с помощью функций MPI.COMM_WORLD.Send и MPI.COMM_WORLD.Recv обменивается ai и bi, вычисляя свои ci.

Второе приложение получает на вход произвольный вектор параметром, после чего делит его между процессами равномерно (если остались части вектора, которые не удалось равномерно распределить между процессами – они передаются на обработку последнему процессу сверх уже выделенной части).

```
PS C:\Users\npc\Desktop\parallel-labs> java -jar ".\mpj-v0_44\lib\starter.jar" -np 4 -cp ".\target\classes" neko.lab2.part1
MPJ Express (0.44) is started in the multicore configuration
P0 (id=21): a0=0, b0=2
P3 (id=22): a3=3, b3=8
P1 (id=24): a1=1, b1=4
P2 (id=23): a2=2, b2=6
P1 (id=24): c1=3+2=5
P0 (id=21): c0=1+6=7
P3 (id=22): c3=2+4=6
P2 (id=23): c2=0+8=8
```

Рисунок 2 – Результат работы программы part1

```
PS C:\Users\npc\Desktop\parallel-labs> java -jar ".\mpj-v0_44\lib\starter.jar" -np 4 -Dvector="5 5 -5" -cp ".\target\classes" neko.lab2.part2
MPJ Express (0.44) is started in the multicore configuration
Sum: 5
```

Рисунок 3 – Результат работы программы part2 (вектор 5 5 -5)

```
PS C:\Users\npc\Desktop\parallel-labs> java -jar ".\mpj-v0_44\lib\starter.jar" -np 4 -Dvector="5 5 4" -cp ".\target\classes" neko.lab2.part2
MPJ Express (0.44) is started in the multicore configuration
Sum: 10
```

Рисунок 4 – Результат работы программы part2 (вектор 5 5 4)

Код программы (lab2.part1.java)

```
package neko.lab2;

import mpi.MPI;

public class part1 {
    public static void main(String[] args) {
        MPI.Init(args);

        int rank = MPI.COMM_WORLD.Rank();

        int ai;
        ai = rank;
        int bi = (rank + 1) * 2;

        System.out.println("P" + rank + " (id=" + Thread.currentThread().getId()
                + "): a" + rank + "=" + ai + ", b" + rank + "=" + bi);

        int[] received_a = new int[1];
        int[] received_b = new int[1];

        int send_a_to, send_b_to;
        int get_a_from, get_b_from;

        switch (rank) {
            case 0:
                send_a_to = 2; // send to solve c2
                send_b_to = 1; // send to solve c1
                get_a_from = 1; // get from c1
                get_b_from = 2; // get from c2
                break;
            case 1:
                send_a_to = 0;
                send_b_to = 3;
                get_a_from = 3;
                get_b_from = 0;
                break;
            case 2:
```

```
                send_a_to = 3;
                send_b_to = 0;
                get_a_from = 0;
                get_b_from = 3;
                break;
            case 3:
                send_a_to = 1;
                send_b_to = 2;
                get_a_from = 2;
                get_b_from = 1;
                break;
            default:
                throw new IllegalStateException("idk count of ranks");
        }

        if (rank == 0 || rank == 3) {
            MPI.COMM_WORLD.Send(new int[]{ai}, 0, 1, MPI.INT, send_a_to, 99);
            MPI.COMM_WORLD.Send(new int[]{bi}, 0, 1, MPI.INT, send_b_to, 99);
            MPI.COMM_WORLD.Recv(received_a, 0, 1, MPI.INT, get_a_from, 99);
            MPI.COMM_WORLD.Recv(received_b, 0, 1, MPI.INT, get_b_from, 99);
        } else {
            MPI.COMM_WORLD.Recv(received_a, 0, 1, MPI.INT, get_a_from, 99);
            MPI.COMM_WORLD.Recv(received_b, 0, 1, MPI.INT, get_b_from, 99);
            MPI.COMM_WORLD.Send(new int[]{ai}, 0, 1, MPI.INT, send_a_to, 99);
            MPI.COMM_WORLD.Send(new int[]{bi}, 0, 1, MPI.INT, send_b_to, 99);
        }

        System.out.println("P" + rank + " (id=" + Thread.currentThread().getId()
                + "): c" + rank + "=" + received_a[0] + "+" + received_b[0] + "=" +
(received_a[0] + received_b[0]));

        MPI.Finalize();
    }
}
```

Код программы (lab2.part2.java)

```
package neko.lab2;

import mpi.MPI;

public class part2 {

    public static void main(String[] args) {
        MPI.Init(args);

        int np = MPI.COMM_WORLD.Size();
        int rank = MPI.COMM_WORLD.Rank();

        // Get vector from command
        String vectorString = System.getProperty("vector", "1 2 3 4 5");
        String[] elements = vectorString.split("\\s+");
        int[] vector = new int[elements.length];
        for (int i = 0; i < elements.length; i++) {
            vector[i] = Integer.parseInt(elements[i]);
        }

        // Size of input vector
        int size = vector.length;

        // Resolve size of local vectors
        int blockSize = size / np;
```

```
        int remainder = size % np;

        // Size of local vectors
        int localSize = blockSize + (rank == np - 1 ? remainder : 0);
        int[] localVector = new int[localSize];

        if (rank == 0) {
            // P0 send parts of vector
            int offset = 0;
            for (int i = 0; i < np; i++) {
                int count = blockSize + (i == np - 1 ? remainder : 0);
                if (i == 0) {
                    // P0 get local vector
                    System.arraycopy(vector, offset, localVector, 0, count);
                } else {
                    // Send vectors to other P's
                    MPI.COMM_WORLD.Send(vector, offset, count, MPI.INT, i, 0);
                }
                offset += count;
            }
        } else {
            // Other P's get local vector
            MPI.COMM_WORLD.Recv(localVector, 0, localSize, MPI.INT, 0, 0);
        }

        // Sum inside P
        int localSum = 0;
        for (int j : localVector) {
            if (j % 2 != 0) {
                localSum += j;
            }
        }

        // Get sum from other P
        int[] globalSums = new int[np];
        MPI.COMM_WORLD.Gather(new int[]{localSum}, 0, 1, MPI.INT, globalSums, 0, 1,
MPI.INT, 0);

        // Sum in P0
        if (rank == 0) {
            int totalSum = 0;
            for (int sum : globalSums) {
                totalSum += sum;
            }
            System.out.println("Sum: " + totalSum);
        }

        MPI.Finalize();
    }
}
```

Вывод

В ходе выполнения лабораторной работы были изучены основы работы с библиотекой MPI (Message Passing Interface) для организации параллельных вычислений в Java (MPJ). Были реализованы три программы, каждая из которых постепенно демонстрирует все более глубокие аспекты использования MPI для распределения задач между процессами.

Выполнение лабораторной работы №1 позволило ознакомиться с базовыми принципами работы MPI и создать простое приложение для параллельных вычислений, которые никак не пересекались между собой.

Во время выполнения задания 1 лабораторной работы №2 можно было ознакомиться с новыми механизмами взаимодействия между процессами, что позволяло передать данные между ними и использовать их в конечном результате. Задание 2 было нацелено на дальнейшее использование этой связи, так как мы смогли с помощью одного процесса распределить задания, а после окончания вычислений объединить результаты процессов, сделав итоговый вывод от одного процесса, а не от разных вразнобой.