ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Факультет информационных систем и технологий
Кафедра программного обеспечения и управления в технических системах

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине Параллельное программирование
название (при наличии)

Вычисление числа π
название работы (при наличии)

ВЫПОЛНИЛ
студент гр. ПО–61 Булычев И. Д.
(группа) (ФИО)

ПРОВЕРИЛА
к. т. н., доцент Мезенцева Е. М.
(должность) (ФИО)

Самара
2019

1 Цель лабораторной работы

1.1 Цель работы

Разработать и запустить на кластере программу на любом из языков программирования, вычисляющую число π.

1.2 Используемое программное обеспечение

Для выполнения лабораторной работы мною было использовано следующее программное обеспечение:

- ОС Ubuntu 18.10
- IDE Intellij Idea 2018.3
- JDK 1.8
- MPJ Express 0.44

2 Ход выполнения лабораторной работы

2.1 Формула расчета

Для расчета числа π используется формула Бэйли-Боруэйна-Плаффа (ББП-формула):

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{1}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right). \quad (1)$$

2.2 Реализация вычислений

Для получения числа π с высокой точностью все вычисления проводятся с помощью Java-класса BigDecimal.

Для использования данного класса необходимо создать пользовательскую MPI-функцию (интерфейс MPI.User_function) и MPI-операцию (класс mpi.Op) для использования в функции MPI.COMM_WORLD.Reduce().

Входные данные для вычислений (количество членов ряда, количество знаков после запятой) указано в файле input.txt.

2.2.1 Создание пользовательской функции

Так как тип User_function является интерфейсным, то при создании объекта данного типа необходимо переопределить метод Call, принимающий в качестве аргументов:

- входной массив значений;
- индекс массива, начиная с которого будет применяться данная функция во входном массиве;
- выходной массив значений, в который записывается результат;
- индекс массива, начиная с которого будет записываться результат в выходном массиве;
- количество элементов массивов, с которыми нужно произвести действия;
- тип данных, в который преобразуется выходной массив.

В методе Call описывается операция, которая должна выполняться

на соответствующих элементах входного и выходного массива с записью результата в выходной массив. В данном случае выполняется операция сложения двух чисел типа BigDecimal.

2.2.2 Создание пользовательской операции

Для использования функции Reduce с пользовательскими объектами, необходимо создать пользовательскую операцию. Для этого нужно создать объект типа mpi.Op.

Параметры конструктора:

- Объект пользовательской функции;
- Логическое значение — является ли функция коммутативной.

2.2.3 Функция вычисления числа π

Число π вычисляет функция calculatePi(). Эта функция использует формулу Бэйли-Боруэйна-Плаффа. Её входные параметры:

- Начальный индекс суммы;
- Конечный индекс суммы;
- Количество знаков после запятой.

Данная функция вычисляется параллельно. Каждый процесс вычисляет сумму указанного диапазона членов ряда.

2.2.4 Получение конечного результата

Для получения конечного результата вычислений — редуцирования результатов вычислений каждого процесса — используется функция Reduce(). Ее входные параметры:

- Входной массив результатов вычисления данного процесса:
- Начальный индекс входного массива;
- Выходной массив результатов редуцирования;
- Начальный индекс выходного массива;
- Количество элементов, над которыми производится редуцирование;
- Тип элементов;
- Операция, производимая над соответствующими элементами входного и выходного массивов с записью в выходной массив;

- Номер процесса, в котором будет храниться финальное значение переменной.

Актуальное (финальное) значение числа π хранится в переменной процесса 0. В остальных процессах данная переменная так же присутствует, но в ней хранятся промежуточные результаты суммирования. Поэтому для вывода числа π на экран запрашивается значение переменной процесса 0.

3 Результаты выполнения лабораторной работы

3.1 Листинг приложения

```
package lab3;

import mpi.*;

import java.io.File;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;

public class PiCalc {
    public static void main(String[] args) throws Exception {

        Scanner in = new Scanner(new File("src/lab3/input.txt"));
        int scale = in.nextInt(); // precision
        int iter = in.nextInt(); // iteration num

        // o is an input array, o1 is a reduce array (accumulator)
        User_function function = new User_function() {
            @Override
            public void Call(Object o, int i, Object o1, int i1, int i2,
    Datatype datatype) throws MPIException {
                BigDecimal arg = ((BigDecimal[]) o)[0];
                Object[] resultArray = ((Object[]) o1);
                BigDecimal accumulator = (BigDecimal) resultArray[0];
                resultArray[0] = accumulator.add(arg);
            }
        };
        //
        user defined reduce operation
        Op op = new Op(function, false);
        BigDecimal[] res = {
            BigDecimal.ZERO
        };
        MPI
        .Init(args);
        int size = MPI.COMM_WORLD.Size(); // proc amnt
        int rank = MPI.COMM_WORLD.Rank(); // proc num

        // just for equal amount of iterations for each process (like (170 /
    4) * 4 == 168 )
        iter = ( iter / size ) * size;
        int processStep = iter / size; // iter amnt for each proc
        BigDecimal[] stepRes = {
            BigDecimal.ZERO
        }; //
```

```
        start from zero
        stepRes[0] = calculatePi(rank*processStep, (rank+1) * processStep,
    scale); // calculating for each process
        MPI.COMM_WORLD.Reduce(stepRes, 0, res, 0, 1, MPI.OBJECT, op, 0); //
    sum all in res
        MPI.Finalize(); // shutting down
        if (rank == 0) {
            System.out.println(res[0]);
            System.out.println(res[0].equals(calculatePi(0, iter, scale))); //
     correct check
        }
    }

    // Bailey-Borwein-Plouffe formula
    private static BigDecimal calculatePi(int start, int end, int scale) {
        BigDecimal e = BigDecimal.ZERO;
        for (int k = start; k < end; k++) {
            BigDecimal a0 = new BigDecimal(16).pow(k);
            BigDecimal a1 = new BigDecimal(4).divide(new BigDecimal(8*k+1),
    scale, RoundingMode.HALF_UP);
            BigDecimal a2 = new BigDecimal(2).divide(new BigDecimal(8*k+4),
    scale, RoundingMode.HALF_UP);
            BigDecimal a3 = new BigDecimal(1).divide(new BigDecimal(8*k+5),
    scale, RoundingMode.HALF_UP);
            BigDecimal a4 = new BigDecimal(1).divide(new BigDecimal(8*k+6),
    scale, RoundingMode.HALF_UP);
            BigDecimal a5 = a1.subtract(a2).subtract(a3).subtract(a4);
            BigDecimal a6 = BigDecimal.ONE.divide(a0, scale, RoundingMode.
    HALF_UP);
            BigDecimal elem = a5.multiply(a6);
            e = e.add(elem);
        }
        return e;
    }
}
```

3.2 Результат выполнения

Входные значения равны 200 и 100 (итерации и точность).

```
3.14159265358979323846264338327950288419716939937510582097494459230781
64062862089986280348253421170679821480865132823066470938350119627942072033649
93689306529186774810464437472757458356894967998305685447223493781296568394826
65985052588607165729859873368906980745908846334436029838283287920823143642933
14344581896143365041340636074918524482183533734830644393147695032526939304802
74857613409186988719559717 2
    true
```

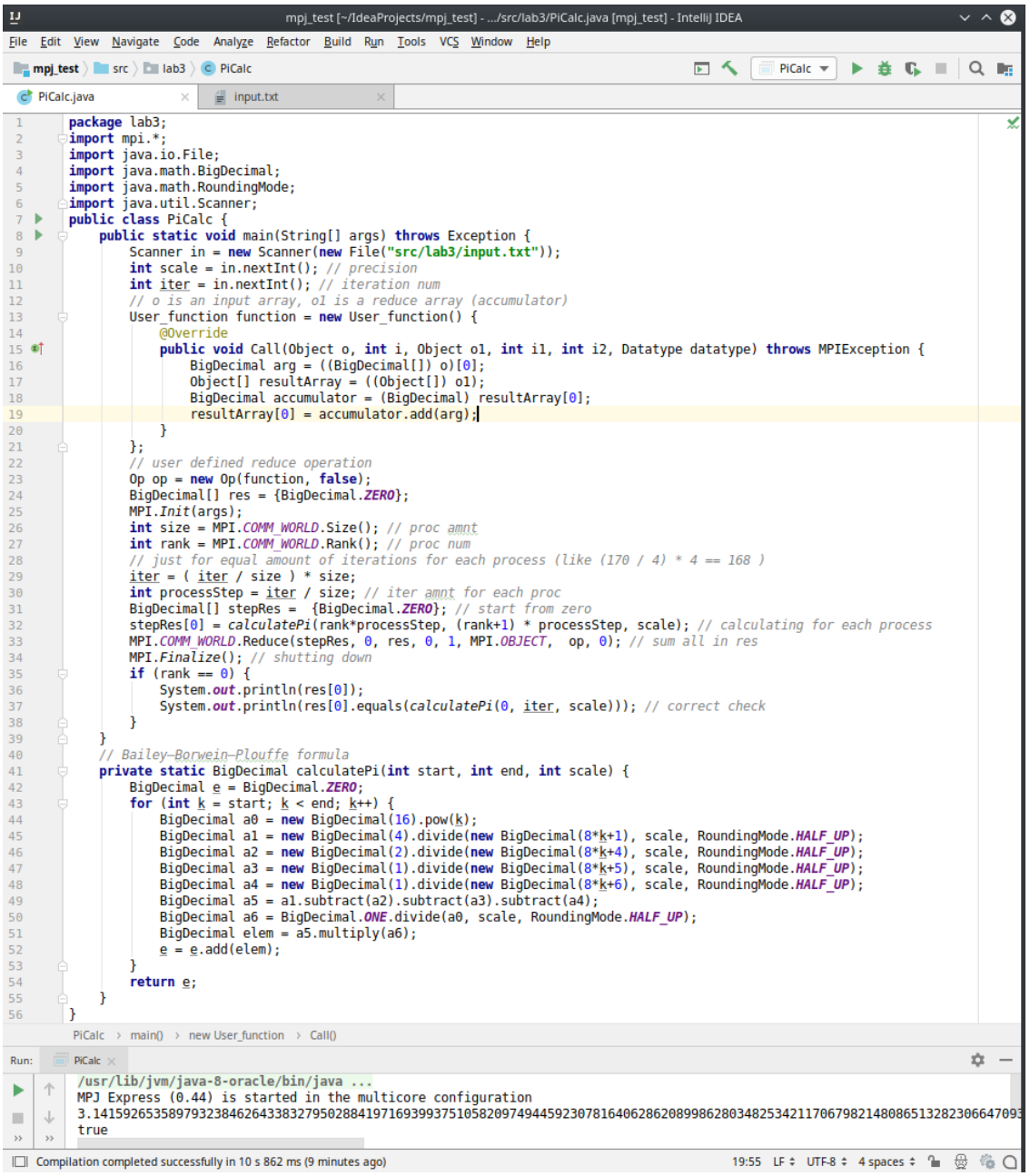

Рис. 3.1 — Листинг и результат выполнения

4 Выводы по результатам выполнения лабораторной работы

В ходе работы была написано консольное приложение на языке Java в среде программирования Intellij Idea, к которой подключена библиотека MPJ Express. Каждый процесс рассчитывает свои значения суммы членов ряда в заданном диапазоне, после чего выполняется суммирование промежуточных результатов с помощью функции Reduce. После чего на экран выводится значение числа π и проверка корректности результата — сравнение с числом π, вычисленным не параллельно.

Выполнен тестовый запуск программы на кластере, состоящем из ядер процессора компьютера.