Creating MapReduce program to calculating pi

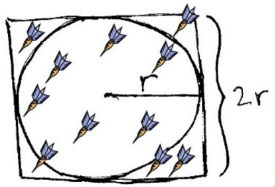
Table of Content

1. Theory
2. Setup
3. Create input files
4. Execution
5. Result
6. Conclusion
7. References

Theory

There are many ways to calculate Pi. But in this project, we are using MapReduce

- Throw N darts on the board. Each dart lands at a random position (x,y) on the board.
  - Note if each dart landed inside the circle or not
    - Check if $x^2+y^2<r$
  - Take the total number of darts that landed in the circle as S



Title

$$4\left(\frac{S}{N}\right)=\pi$$

Formula:

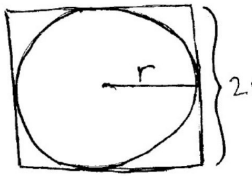
4 * S / N = 4 * (pi * r * r) / (4 * r * r) = pi

Note:

- **S = darts inside the circle = the area of the circle**
- **N = darts on the board = the area of the square**

Sample MapReduce Code- Estimate π

- Estimating π by random sampling
- Imagine you have a dart board like so:



$$P(dart\ in\ circle)=\frac{\pi r^2}{4r^2}=\frac{\pi}{4}$$

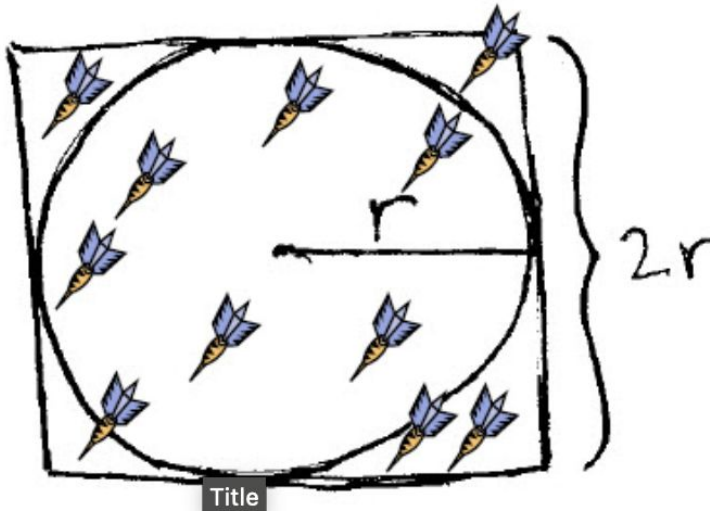
$$\Rightarrow \pi = 4P(dart\ in\ circle)$$

- π is simply the (ratio of darts that land inside the circle to the total number of darts thrown) times 4

How?

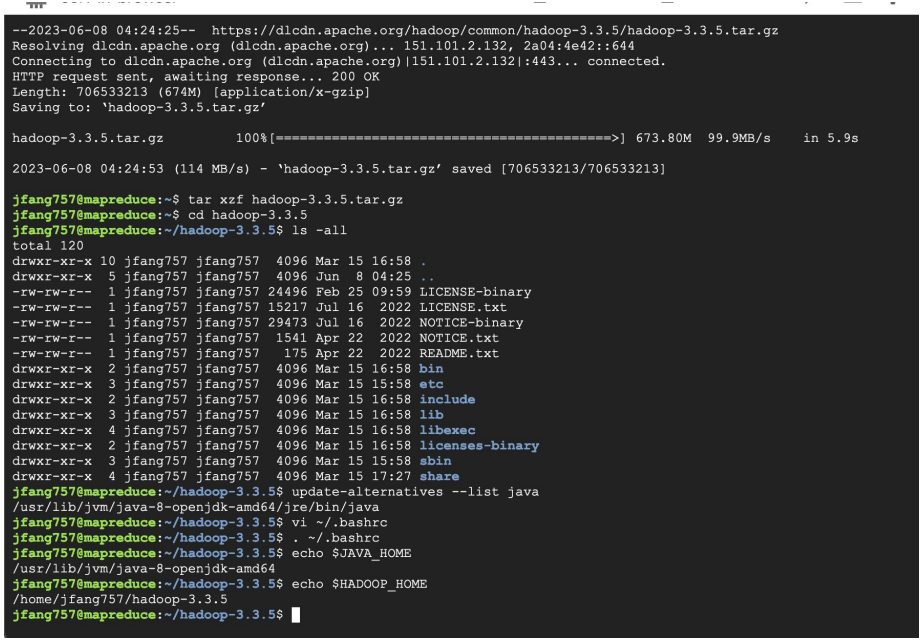
1. Let (x,y) be a random position of the dart inside the square. Then, we map each (x,y) pari to a result. If the pair is inside the circle, then result =1, otherwise 0.
2. To calculate the Pi, we need to sum all the pair result inside the circle as S, and divide by the total number of pair N, multiply by 4, and get Pi.

   Pi = 4(S/N)



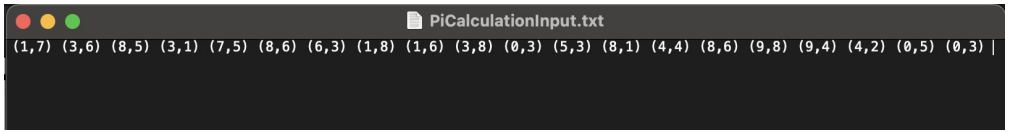
Setup

Follow the standard setup for Hadoop in GCP Ubuntu



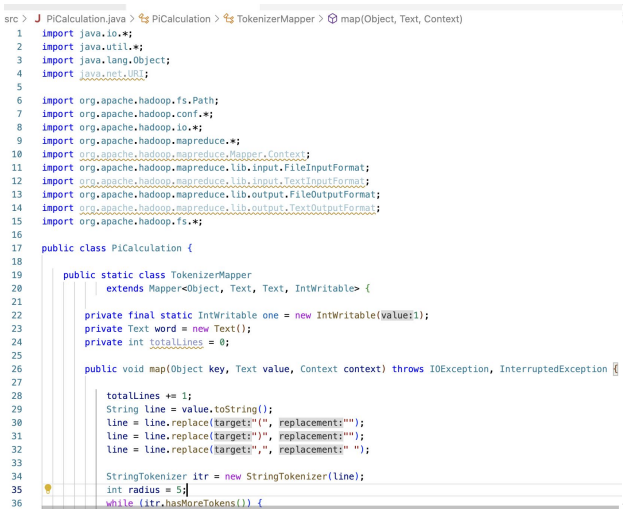
Create input files

Run a java program to create a file contain random number pair.




```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
public class GenerateNums{
    Run | Debug
    public static void main(String[] args){
        System.out.println(x:"How many random numbers to generate: ");
        Scanner input = new Scanner(System. in);
        int RandomNumCount = input.nextInt();

        System.out.println(x:"What's the radium number? ");
        int radius = input.nextInt();
        int diameter = radius * 2;
        input.close();

        try{
            File file = new File(pathname:"./PiCalculationInput.txt");
            file.createNewFile();

            FileWriter writer = new FileWriter(file);

            for(int i = 0; i < RandomNumCount; i++){
                int xvalue = (int) (Math.random() * diameter);
                int yvalue = (int) (Math.random() * diameter);
                writer.write("(" + xvalue + "," + yvalue + ") ");
                System.out.print("(" + xvalue + "," + yvalue + ") ");
            }
            writer.flush();
            writer.close();
        }catch(IOException e){
            e.printStackTrace();
        }
    }
}
```

Execution-Pi MapReduce program



```
src > J PiCalculation.java > PiCalculation > TokenizerMapper > map(Object, Text, Context)
1  import java.io.*;
2  import java.util.*;
3  import java.lang.Object;
4  import java.net.URI;
5
6  import org.apache.hadoop.fs.Path;
7  import org.apache.hadoop.conf.*;
8  import org.apache.hadoop.io.*;
9  import org.apache.hadoop.mapreduce.*;
10 import org.apache.hadoop.mapreduce.Mapper.Context;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
15 import org.apache.hadoop.fs.*;
16
17 public class PiCalculation {
18
19     public static class TokenizerMapper
20             extends Mapper<Object, Text, Text, IntWritable> {
21
22         private final static IntWritable one = new IntWritable(value:1);
23         private Text word = new Text();
24         private int totalLines = 0;
25
26         public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
27
28             totalLines += 1;
29             String line = value.toString();
30             line = line.replace(target:"(", replacement:"");
31             line = line.replace(target:")", replacement:"");
32             line = line.replace(target:",", replacement:" ");
33
34             StringTokenizer itr = new StringTokenizer(line);
35             int radius = 5;
36             while (itr.hasMoreTokens()) {
```

```
src > J PiCalculation.java > PiCalculation > TokenizerMapper > map(Object, Text, Context)
33
34             StringTokenizer itr = new StringTokenizer(line);
35             int radius = 5;
36             while (itr.hasMoreTokens()) {
37                 String x, y;
38                 x = itr.nextToken();
39
40                 if (itr.hasMoreTokens()) {
41                     y = itr.nextToken();
42                 } else {
43                     y = "0";
44                 }
45
46                 int xvalue = (int) (Integer.parseInt(x));
47                 int yvalue = (int) (Integer.parseInt(y));
48                 double check = Math.sqrt(Math.pow((radius - xvalue), 2) + Math.pow((radius - yvalue), 2));
49
50                 if (check < radius) {
51                     word.set(string:"inside");
52                 } else {
53                     word.set(string:"outside");
54                 }
55                 context.write(word, one);
56             }
57         }
58     }
59
60     public static class IntSumReducer
61             extends Reducer<Text, IntWritable, Text,
62         private IntWritable result = new IntWritable(
63
64         public void reduce(Text key, Iterable<IntWritable> values,
65                 Context context) throws IOException, InterruptedException {
66             int sum = 0;
67             for (IntWritable val : values) {
68                 sum += val.get();
69             }
70             result.set(sum);
71             context.write(key, result);
```

Iterable<IntWritable> values - PiCalculation.IntSumReduc
Iterable<IntWritable>, Context)

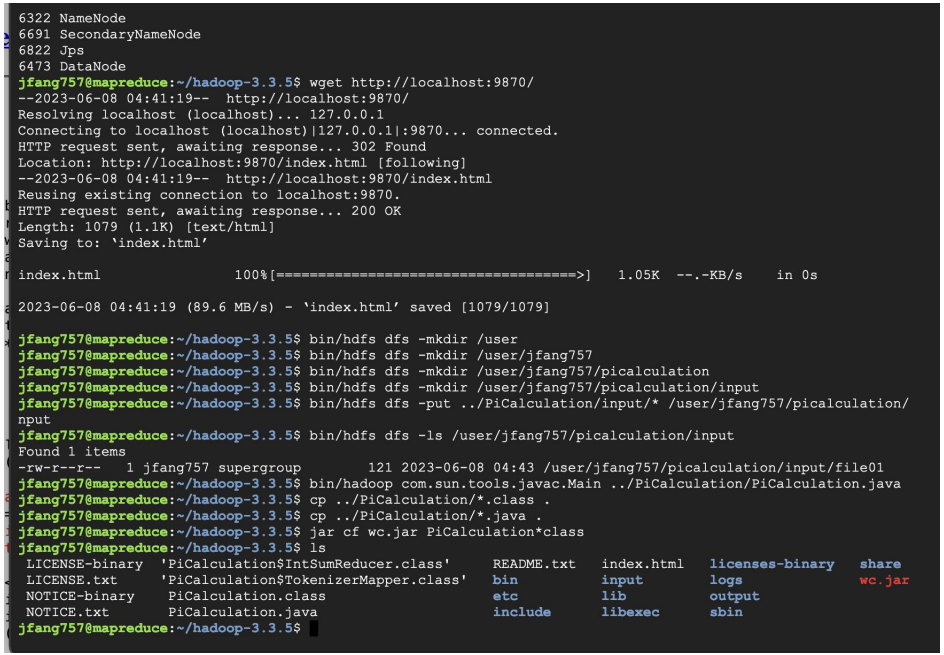
```
71                 context.write(key, result);
72             }
73         }
74
Run | Debug
75         public static void main(String[] args) throws Exception {
76             Configuration conf = new Configuration();
77             Job job = Job.getInstance(conf, jobName:"pi calculation");
78             job.setJarByClass(cls:PiCalculation.class);
79             job.setMapperClass(cls:TokenizerMapper.class);
80             job.setCombinerClass(cls:IntSumReducer.class);
81             job.setReducerClass(cls:IntSumReducer.class);
82             job.setOutputKeyClass(theClass:Text.class);
83             job.setOutputValueClass(theClass:IntWritable.class);
84             FileInputFormat.addInputPath(job, new Path(args[0]));
85             FileOutputFormat.setOutputPath(job, new Path(args[1]));
86             // System.exit(job.waitForCompletion(true) ? 0 : 1);
87             job.waitForCompletion(verbose:true);
88             String filePath = args[1] + "/" + "part-r-00000";
89             Path path = new Path(filePath);
90             FileSystem fs = FileSystem.get(path.toUri(), conf);
91
92             BufferedReader br = new BufferedReader(new InputStreamReader(fs.open(path)));
93
94             String z, inside = null, outside = null;
95
96             String line1, line2;
96             String line1, line2;
97
98             line1 = br.readLine();
99             System.out.println(line1);
100            line2 = br.readLine();
101            System.out.println(line2);
102
103            line1 = line1.replace(target:"inside", replacement:"").trim();
104            line2 = line2.replace(target:"outside", replacement:"").trim();
105
106            System.out.println("Inside:" + line1 + ", Outside:" + line2);
107
108            if (line1 != null && line2 != null) {
109                double invalue = Double.valueOf(line1);
110                double outvalue = Double.valueOf(line2);
111                double pi = 4 * (invalue / (invalue + outvalue));
112                System.out.println("PI:" + pi);
113            }
114
115            fs.close();
116        }
117    }
```

Execution

Make the HDFS directories required to execute MapReduce jobs

Copy the input files into the distributed file system

Move .class files to hadoop-3.3.5 directory and create jar



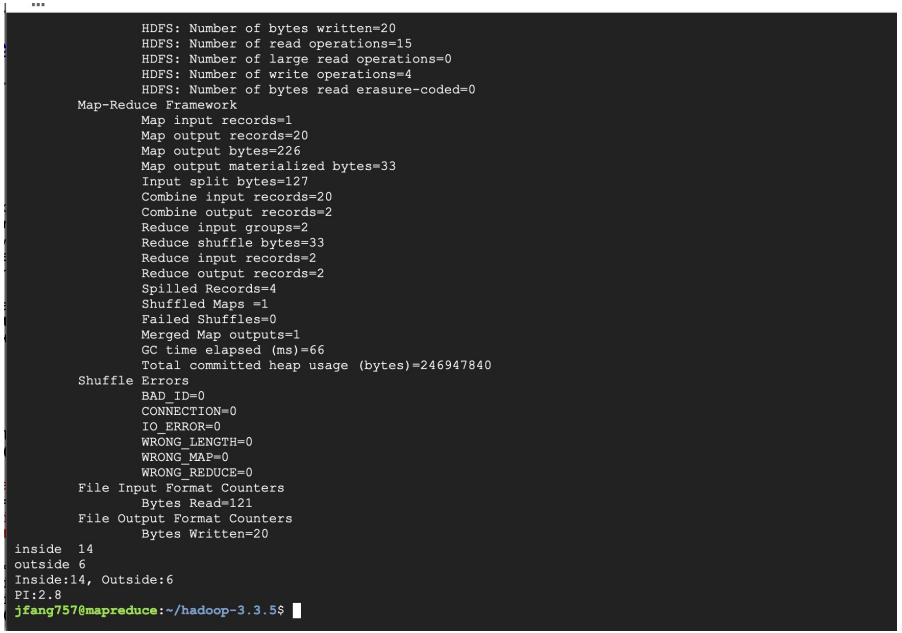
Result

I use 20 pairs with a radius of 5 for this project, and the results is

Inside 14

Outside 6

Pi 2.8



Conclusion

The result 2.8 is far off Pi, but I only use 20 pairs of numbers. If we increase the number of pairs to 200 or more, the result will be much closer to the Pi.

References

Exercises for Pi: https://hc.labnet.sfbu.edu/~henry/npu/classes/mapreduce/pi/slide/exercise_pi.html

Sample code:
https://hc.labnet.sfbu.edu/~henry/npu/classes/mapreduce/pi/hw/q1/2022_fall/PiCalculation.html

MapRedcue Pi concept:
https://hc.labnet.sfbu.edu/~henry/npu/classes/mapreduce/pi/slide/mapreduce_pi.html