Creating MapReduce program to calculating Pi

Emily Weng

Overview

- Introduction
- Design
- Implementation
- Test
- Enhancement Ideas
- Conclusion
- Reference

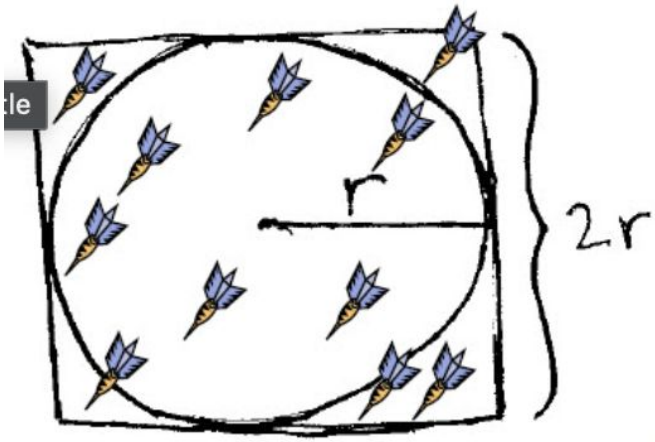
Introduction

- π is a mathematical constant, approximately equal to 3.14159.
- Our target is to use MapReduce and calculate Pi
- MapReduce is a programming model that is used for processing and generating big data sets

Design

How can we calculate Pi by throwing darts?

- Concept:
  - Throw **N** darts on the circle
  - See if the darts landed in the circle or not
  - Take the amount of darts in the circle as **S**
- Check if **x^2 + y^2 < r**
- **Formula: 4 * S / N = 4 * (pi * r * r) / (4 * r * r) = pi**



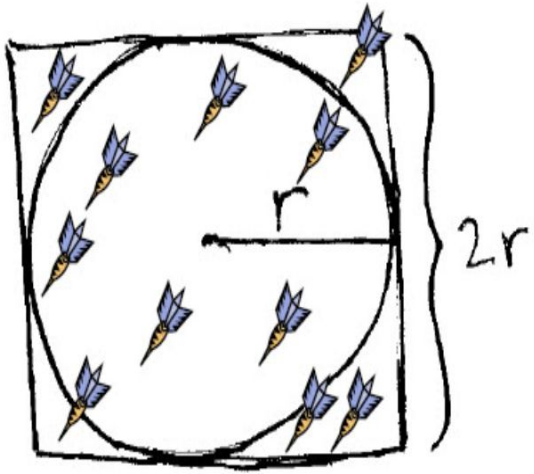
Design

How to check if the darts are inside or outside?

- (x - center_x)^2 + (x - center_x)^2 compare r^2
- Inside: if it is < than r^2
- Outside: if it is > than r^2
- On the circle: if it is = r^2

Example:

- If radius is 4, then based on the picture below, we can calculate
- pi = 4 * (S / N)
- S = 6, N = 10
- pi = 4 (6/10)
- pi = 4(⅗)
- pi = 4(0.6) = 2.4
- So pi is 2.4



Implementation - Environment

We'll be using GCP, and setting up with ubuntu and Hadoop

```
eweng909@ubuntu:~$ ls
hadoop-3.3.5.tar.gz
eweng909@ubuntu:~$ tar xzf hadoop-3.3.5.tar.gz
eweng909@ubuntu:~$ ls
hadoop-3.3.5  hadoop-3.3.5.tar.gz
eweng909@ubuntu:~$ cd hadoop-3.3.5
```

Implementation

Step 1.1: Generate an input file to the Pi MapReduce program

- Create a file to generate random numbers
- Then compile the java program

```
eweng909@ubuntu1:~/PiCalculation$ vi GenerateRandomNumbers.java
eweng909@ubuntu1:~/PiCalculation$ javac GenerateRandomNumbers.java
eweng909@ubuntu1:~/PiCalculation$ java GenerateRandomNumbers
```

```java
import java.util.Scanner;

public class GenerateRandomNumbers {

        public static void main(String[] args) {
                System.out.println("How many random numbers to generate:");
                Scanner input =new Scanner(System.in);
                int RandomNumCount = input.nextInt();

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

                int num []= new int[RandomNumCount] ;
    for (int i=0; i<RandomNumCount; i++) {
     num [i]= (int) (Math.random()* diameter);
     System.out.print(num[i]+" ");
                }
        }}
~
```

Implementation

Generate the Random number and Save as input file for Hadoop Job

```
eweng909@ubuntu1:~/PiCalculation$ java GenerateRandomNumbers
How many random numbers to generate:
100
What's the radius number?
4
7 6 4 0 6 5 2 1 1 6 1 2 6 3 1 7 0 0 1 1 5 2 1 5 0 3 3 2 7 5 7 7 0 6 2 7 5 7 1 5 0 2 7 1 0 6 3 4 1 1 4 3 6 0 0 7
1 7 6 1 4 4 2 4 7 2 0 4 0 4 1 7 3 0 4 7 4 5 7 0 3 6 2 5 7 6 5 3 0 6 3 2 7 2 3 4 4 1 3 2 eweng909@ubuntu1:~/PiCal
culation$
```

Implementation

Create PiCalculation.java file to perform MapReduce Operation(Radius=4)

Map function()

```
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    private int totalLines = 0;

    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        totalLines += 1;
        String line = value.toString();
        line = line.replace("(", "");
        line = line.replace(")", "");
        line = line.replace(",", " ");

        StringTokenizer itr = new StringTokenizer(line);
        int radius = 200; // Same as the one you give in PiDataGenerator stage
        while (itr.hasMoreTokens()) {
            String x, y;
            x = itr.nextToken();

            if (itr.hasMoreTokens()) {
                y = itr.nextToken();
            } else {
                y = "0";
            }

            int xvalue = Integer.parseInt(x);
            int yvalue = Integer.parseInt(y);
            double check = Math.sqrt(Math.pow((radius - xvalue), 2) + Math.pow((radius - yvalue), 2));

            if (check < radius) {
                word.set("inside");
            } else {
                word.set("outside");
            }
            context.write(word, one);
        }
    }
```

Implementation

Create PiCalculation.java file to perform MapReduce Operation(Radius=4)

Reduce function

```
    public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, Interrup
tedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }
```

Implementation

Create PiCalculation.java file to perform MapReduce Operation(Radius=4)

Main function()

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "pi calculation");
    job.setJarByClass(PiCalculation.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.waitForCompletion(true);

    String filePath = args[1] + "/" + "part-r-00000";
    Path path = new Path(filePath);
    FileSystem fs = FileSystem.get(path.toUri(), conf);

    BufferedReader br = new BufferedReader(new InputStreamReader(fs.open(path)));

    String line1 = br.readLine();
    System.out.println(line1);
    String line2 = br.readLine();
    System.out.println(line2);

    line1 = line1.replace("inside", "").trim();
    line2 = line2.replace("outside", "").trim();

    System.out.println("Inside:" + line1 + ", Outside:" + line2);

    if (line1 != null && line2 != null) {
        double invalue = Double.valueOf(line1);
        double outvalue = Double.valueOf(line2);
        double pi = 4 * (invalue / (invalue + outvalue));
        System.out.println("PI:" + pi);
    }

    fs.close();
}
```

Commands

Go to hadoop directory:

cd hadoop-3.3.5

sbin/start-dfs.sh : run the daemons

Create HDFS Directories:

bin/hdfs dfs -mkdir /user

bin/hdfs dfs -mkdir /user/eweng909

bin/hdfs dfs -mkdir /user/eweng909/picalculation

bin/hdfs dfs -mkdir /user/eweng909/picalculation/input

Command:

Copy the input file into the distributed filesystem:

- bin/hdfs dfs -put ../PiCalculation/input/* /user/eweng909/picalculation/input

Run the jar file for PiCalculation.java

- ../bin/hadoop com.sun.tools.javac.Main ./PiCalculation.java
- jar cf wc.jar ./PiCalculation*.class
- ../bin/hadoop jar wc.jar PiCalculation /user/eweng909/picalculation/input /user/eweng909/picalculation/output

Current Results

```
eweng909@ubuntu1:~/hadoop-3.3.5$ bin/hdfs dfs -ls /user/eweng909/picalculation/output
Found 2 items
-rw-r--r--   1 eweng909 supergroup          0 2024-06-05 17:28 /user/eweng909/picalculation/output/_SUCCESS
-rw-r--r--   1 eweng909 supergroup          0 2024-06-05 17:28 /user/eweng909/picalculation/output/part-r-00000
```

```
eweng909@ubuntu1:~/hadoop-3.3.5$ cd PiCalculation
eweng909@ubuntu1:~/hadoop-3.3.5/PiCalculation$ ../bin/hadoop fs -cat /user/eweng909/picalculation/output/part-r-
00000
eweng909@ubuntu1:~/hadoop-3.3.5/PiCalculation$
```