Practical - 6: Analyse impact of different number of mapper and reducer on same definition as practical 4.

- Prepare a conclusive report on the analysis.

By Divya Mahur (18bce106)

➔ **`java` file**

```
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.*;
import java.time.Duration;
import java.time.Instant;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class Prac_6 {

  public static class Practical_6_Mapper_1 extends Mapper<Object,
Text, Text, Text>{

    public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {

      Text word = new Text();

      StringTokenizer itr = new StringTokenizer(value.toString());

      while (itr.hasMoreTokens()) {

        word.set(itr.nextToken());

        context.write(new Text("All"), word);

      }

    }

  }


  public static class Practical_6_Reducer_1 extends
Reducer<Text,Text,Text,Text> {

    private int count = 0;

    private int sum = 0;

    private int max = 0;

    public void reduce(Text key, Iterable<Text> value, Context
context) throws IOException, InterruptedException{


  for(Text i: value){

    int n = Integer.parseInt(i.toString());

    sum+=n;

    count++;

                if( n > max )

                    max = n;

  }
```

```
  double avg = (double)sum/count;

  context.write(new Text("The largest integer from the data:"), new
Text(Integer.toString(max)));

        context.write(new Text("The average of the data: "), new
Text(Double.toString(avg)));

    }

  }


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

    long start1 = System.nanoTime();

    Configuration conf = new Configuration();

    Job job = Job.getInstance(conf, "Prac 6");

    job.setJarByClass(Prac_6.class);

    job.setMapperClass(Practical_6_Mapper_1.class);

    job.setReducerClass(Practical_6_Reducer_1.class);

    job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(Text.class);

    job.setNumReduceTasks(1);

    FileInputFormat.addInputPath(job, new Path(args[0]));

    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);

    long end1 = System.nanoTime();


    System.out.println("Number of Reduce Tasks: " +
job.getNumReduceTasks());

    System.out.println("Elapsed Time in seconds: " +
(end1-start1)/1000000000);

  }

}
```

- **> Plot of analysis with different number of mapper and reducer**

	1	2	3	4	5	6
1	49226572724	46901107258	45327045150	48050998177	63969524583	55732287517
2	52615509409	44859131105	45935858475	50271244656	52343793143	54050332393
3	47997147434	46370667806	49160656001	54398764826	53398635070	56295354154
4	48492911857	47231830217	49051694813	51165121324	54878577223	60532924562
5	49481968467	49510077883	51996412611	54454824084	58437978116	59033430402
6	50267008962	50617173248	51996412611	59434869263	60392935923	64839310160
7	59466097149	53353442402	58419952420	60469203175	62298703017	66138784180
8	55150950808	55454549536	62756416038	62660524247	62646150404	65902702623
9	57528224887	59593964689	62336661384	62351567180	65486299049	69841373320