2CS702 - Big Data Analytics

Practical 4

Aim: **Design MapReduce algorithms to take a very large file of integers and produce as output: a) The largest integer b) The average of all the integers. c) The same set of integers, but with each integer appearing only once.* d) The count of the number of distinct integers in the input.**

Author: Darshil Maru 20BCE514

Guide: Dr. Purnima Gandhi

Distinct WCDriver.java

```java
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {
    public int run(String args[]) throws IOException {
        if (args.length < 2) {
            System.out.println("Please give valid inputs");
            return -1;
        }
        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(IntWritable.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }

    // Main Method
    public static void main(String args[]) throws Exception {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
}
```

Distinct WCMapper.java

```
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {
    public int run(String args[]) throws IOException {
        if (args.length < 2) {
            System.out.println("Please give valid inputs");
            return -1;
        }
        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(IntWritable.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }

    // Main Method
    public static void main(String args[]) throws Exception {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
}
```

Distinct WCReducer.java

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements
Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {
    @Override
    // Reduce Function
    public void reduce(IntWritable key, Iterator<IntWritable> value,
            OutputCollector<IntWritable, IntWritable> output, Reporter
rep)
            throws IOException {
        int counter = 0;
        while (value.hasNext()) {
            IntWritable i = value.next();
            counter++;
            if (counter == 1)
                output.collect(new IntWritable(i.get()), new
IntWritable(i.get()));
        }
    }
}
```

AVG WCDriver.java

```
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
```

```java
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {
    public int run(String args[]) throws IOException {
        if (args.length < 2) {
            System.out.println("Please give valid inputs");
            return -1;
        }
        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(IntWritable.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }

    // Main Method
    public static void main(String args[]) throws Exception {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
}
```

Avg WCMapper.java

```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements
Mapper<LongWritable, Text, IntWritable, IntWritable> {
    @Override
    // Map function
    public void map(LongWritable key, Text value,
            OutputCollector<IntWritable, IntWritable> output, Reporter
rep)
            throws IOException {
        // Splitting the line into spaces
        String data[] = value.toString().split(" ");
        for (String num : data) {
            int number = Integer.parseInt(num);
            output.collect(new IntWritable(1), new IntWritable(number));
        }
    }
}
```

Avg WCReducer.java

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements
Reducer<IntWritable, IntWritable, Text, FloatWritable> {
    @Override
    // Reduce Function
    public void reduce(IntWritable key, Iterator<IntWritable> value,
```

```
            OutputCollector<Text, FloatWritable> output, Reporter rep)
            throws IOException {
        int sum = 0;
        int count = 0;
        // Counting the frequency of each words
        while (value.hasNext()) {
            IntWritable i = value.next();
            sum += i.get();
            count++;
        }
        float sum1 = (float) (sum) / count;
        output.collect(new Text("AVG"), new FloatWritable(sum1));
    }
}
```

Largest WCDriver.Java

```
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {
    public int run(String[] args) throws IOException {
        if (args.length < 2) {
            System.out.println("Please give valid inputs");
            return -1;
        }
        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new Path[] { new Path(args[0])
});
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        conf.setMapperClass(WCMapper.class);
```

```
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(IntWritable.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(IntWritable.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }

    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
}
```

Largest WCMapper.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements
Mapper<LongWritable, Text, IntWritable, IntWritable> {
    public void map(LongWritable key, Text value,
OutputCollector<IntWritable, IntWritable> output, Reporter rep)
            throws IOException {
        String[] data = value.toString().split(" ");
        byte b;
        int i;
        String[] arrayOfString1;
        for (i = (arrayOfString1 = data).length, b = 0; b < i;) {
            String num = arrayOfString1[b];
            int number = Integer.parseInt(num);
```

```
            output.collect(new IntWritable(number), new IntWritable(1));
            b++;
        }
    }
}
```

Largest WCReducer.java

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements
Reducer<IntWritable, IntWritable, IntWritable, IntWritable> {
    int max1 = -1000000000;

    public void reduce(IntWritable key, Iterator<IntWritable> value,
            OutputCollector<IntWritable, IntWritable> output, Reporter
rep) throws IOException {
        this.max1 = Math.max(this.max1, key.get());
        output.collect(key, new IntWritable(this.max1));
    }
}
```