Aayush Shah
19BCE245
27 October 2022

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.

AVERAGE

- **Mapper**

```
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 Mapper<LongWritable, Text, IntWritable,
IntWritable> {
    @Override
    public void map(LongWritable key, Text value, Context ctx) throws
IOException {
        String data[] = value.toString().split(" ");
        for (String num : data) {
            int number = Integer.parseInt(num);
            ctx.write(new IntWritable(1), new IntWritable(number));
        }
} }
```

- **Reducer**

```
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 Reducer<IntWritable, IntWritable, Text,
FloatWritable> {
    @Override
    public void reduce(IntWritable key, Iterator<IntWritable> value,
Context ctx) throws IOException {
       int sum = 0;
       int count=0;
         while (value.hasNext()) {
             IntWritable i = value.next();
             sum += i.get();
             count++;
         }
         float sum1= (float)(sum)/count;
         ctx.write(new Text("AVG"), new FloatWritable(sum1));
} }
```

- **Driver**

```
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 {
    public static void main(String args[]) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "max int");
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(IntWritable.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
} }
```

LARGEST

- **Mapper**

```
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 Mapper<LongWritable, Text, IntWritable,
IntWritable> {
  @override
  public void map(LongWritable key, Text value, Context ctx) 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);
      ctx.write(new IntWritable(number), new IntWritable(1));
      b++;
} }
}
```

- **Reducer**

```
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 Reducer<IntWritable, IntWritable,
IntWritable, IntWritable> {
  int max1 = -1000000000;
  @override
  public void reduce(IntWritable key, Iterator<IntWritable> value, Context
ctx) throws IOException {
    this.max1 = Math.max(this.max1, key.get());
    ctx.write(key, new IntWritable(this.max1));
  }
}
```

- **Driver**

```
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;
```

```java
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class WCDriver {
  public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "average");
      conf.setMapperClass(WCMapper.class);
      conf.setReducerClass(WCReducer.class);
      conf.setMapOutputKeyClass(IntWritable.class);
      conf.setMapOutputValueClass(IntWritable.class);
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      System.exit(job.waitForCompletion(true) ? 0 : 1);
} }
```

DISTINCT

- **Mapper**

```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 Mapper<LongWritable, Text, IntWritable,
IntWritable> {
    @Override
    public void map(LongWritable key, Text value, Context ctx) throws
IOException {
        String data[] = value.toString().split(" ");
        for (String num : data) {
            int number = Integer.parseInt(num);
            ctx.write(new IntWritable(number), new IntWritable(number));
        }
} }
```

- **Reducer**

```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 Reducer<IntWritable, IntWritable,
IntWritable, IntWritable> {
    @Override
    public void reduce(IntWritable key, Iterator<IntWritable> value,
Context ctx) throws IOException {
      int counter=0;
      while (value.hasNext()) {
            IntWritable i = value.next();
            counter++;
            if(counter==1)
              ctx.write(new IntWritable(i.get()), new
IntWritable(i.get()));
        }
} }
```

- **Driver**

```
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 {
    public static void main(String args[]) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "distinct nums");
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(IntWritable.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
} }
```

Conclusion

From this practical, I performed map reduce operations for finding largest, average and distinct element in Hadoop.