Experiment 2: MapReduce Word Count

Source Code:

Create maven project using:

1. mvn -B archetype:generate \
2. -DarchetypeGroupId=org.apache.maven.archetypes \
3. -DgroupId=com.mycompany.app \
4. -DartifactId=my-app

After creating maven project run: mvn clean install

Which will generate:

**pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.sample.app</groupId>  
 <artifactId>WordCount</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <name>WordCount</name>  
 <url>http://maven.apache.org</url>  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-client</artifactId>  
 <version>3.1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.5</version>  
 </dependency>  
 </dependencies>  
 <properties>  
 <maven.compiler.source>1.6</maven.compiler.source>  
 <maven.compiler.target>1.6</maven.compiler.target>  
 </properties>  
  
</project>

**WordCount.java:**

package com.sample.app;  
  
import java.io.IOException;  
import java.util.StringTokenizer;  
  
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 WordCount {  
  
 public static class TokenizerMapper  
 extends Mapper<Object, Text, Text, IntWritable>{  
  
 private final static IntWritable *one* = new IntWritable(1);  
 private Text word = new Text();  
  
 public void map(Object key, Text value, Context context  
 ) throws IOException, InterruptedException {  
 StringTokenizer itr = new StringTokenizer(value.toString());  
 while (itr.hasMoreTokens()) {  
 word.set(itr.nextToken());  
 context.write(word, *one*);  
 }  
 }  
 }  
  
 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, InterruptedException {  
 int sum = 0;  
 for (IntWritable val : values) {  
 sum += val.get();  
 }  
 result.set(sum);  
 context.write(key, result);  
 }  
 }  
  
 public static void main(String[] args) throws Exception {  
 Configuration conf = new Configuration();  
 Job job = Job.*getInstance*(conf, "word count");  
 job.setJarByClass(WordCount.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]));  
 System.*exit*(job.waitForCompletion(true) ? 0 : 1);  
 }  
}

**sample.txt:**

Bhaven Chetan Bhaven Naik Bhaven Naik

Output:

**part-r-00000:**

Bhaven 3  
Chetan 1  
Naik 2

