8CS04-01 Big Data Analytics Lab

Experiment 1: Write a program for execute WordCount Program with MapReduce

WCMapper Java Class file

```
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, Text, IntWritable> {

 // Map function
 public void map(LongWritable key, Text value,
OutputCollector<Text,
 IntWritable> output, Reporter rep) throws IOException
 {

 String line = value.toString();

 // Splitting the line on spaces
 for (String word : line.split(" "))
 {
 if (word.length() > 0)
 {
 output.collect(new Text(word), new IntWritable(1));
 }
 }
 }
}
```

WCReducer Java Class file

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
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<Text,
 IntWritable, Text, IntWritable> {

 // Reduce function
 public void reduce(Text key, Iterator<IntWritable> value,
 OutputCollector<Text, IntWritable> output,
 Reporter rep) throws IOException
 {

 int count = 0;

 // Counting the frequency of each words
 while (value.hasNext())
 {
 IntWritable i = value.next();
 count += i.get();
 }

 output.collect(key, new IntWritable(count));
 }
}
```

WCDriver Java Class file

```
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(Text.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);
	}
}
```

Experiment 2: Write a script to count number of files under specific directory in Hadoop.

```
int count = 0;
FileSystem fs = FileSystem.get(getConf());
boolean recursive = false;
RemoteIterator<LocatedFileStatus> ri = fs.listFiles(new
Path("hdfs://my/path"), recursive);
while (ri.hasNext()){
    count++;
    ri.next();
}
System.out.println("The count is: " + count);
```

Experiment 3: Write a MapReduce code for line count

```
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.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
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 LineCount{
```

```
    public static class LineCntMapper extends Mapper<LongWritable,
Text, Text, IntWritable> {
    Text keyEmit = new Text("Total Lines");
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, Context context){
        try {
            context.write(keyEmit, one);
        }
        catch (IOException e) {
            e.printStackTrace();
            System.exit(0);
        }
        catch (InterruptedException e) {
            e.printStackTrace();
            System.exit(0);
        }
    }
}

    public static class LineCntReducer extends Reducer<Text,
IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context
context){
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        try {
            context.write(key, new IntWritable(sum));
        }
        catch (IOException e) {
            e.printStackTrace();
            System.exit(0);
        }
        catch (InterruptedException e) {
            e.printStackTrace();
            System.exit(0);
        }
    }
}
```

```
    public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "line count2");
    job.setJarByClass(LineCount.class);
    job.setMapperClass(LineCntMapper.class);
    job.setCombinerClass(LineCntReducer.class);
    job.setReducerClass(LineCntReducer.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);
    }
}
```