Software Construction

Lab 11

Usman Noor
BESE-5A
Reg# 32273

I Followed following steps to install hadoop
These steps are exclusive for Mac OS

STEP 1: INSTALL HADOOP

```
$ brew search hadoop
$ brew install hadoop
```

Hadoop will be installed at path /usr/local/Cellar/hadoop

STEP 2: CONFIGURE HADOOP:

Edit hadoop-env.sh, the file can be located at /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop/hadoop-env.sh where 2.6.0 is the hadoop version. Change the line

```
export HADOOP_OPTS="$HADOOP_OPTS -Djava.net.preferIPv4Stack=true"
```

To

```
export HADOOP_OPTS="$HADOOP_OPTS -Djava.net.preferIPv4Stack=true -Djava.security.krb5.realm= -Djava.security.krb5.kdc="
```

Edit Core-site.xml, The file can be located at /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop/core-site.xml add below config

```
<property>
<name>hadoop.tmp.dir</name>
<value>/usr/local/Cellar/hadoop/hdfs/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
```

Edit mapred-site.xml, The file can be located at /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop/mapred-site.xml and by default will be blank add below config

```
<configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:9010</value>
 </property>
</configuration>
```

Edit hdfs-site.xml, The file can be located at /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop/hdfs-site.xml add

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value></value>
 </property>
</configuration>
```

To simplify life edit a ~/.profile and add the following commands. By default ~/.profile might not exist.

```
alias   hstart=<"/usr/local/Cellar/hadoop/2.6.0/sbin/start-dfs.sh;/usr/local/Cellar/hadoop/
2.6.0/sbin/start-yarn.sh">
alias   hstop=<"/usr/local/Cellar/hadoop/2.6.0/sbin/stop-yarn.sh;/usr/local/Cellar/hadoop/
2.6.0/sbin/stop-dfs.sh">
```

and source it

```
$ ssh localhost
$ exit
```

STEP 3: RUN HADOOP

```
$ hstart
$ hstop
```

STEP 4: Wordcount: Create a file called WordCount.java.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.StringTokenizer;

public class WordCount extends Configured implements Tool {
   private final static LongWritable ONE = new LongWritable(1L);

// Mapper Class, Counts words in each line. For each line, break the line into words and
emits them as (word, 1)

public static class MapClass extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable> {
  private final static IntWritable one = new IntWritable(1);
 private Text word = new Text();
```

```
public void map(LongWritable key, Text value,
   OutputCollector<text, intwritable> output,
   Reporter reporter) throws IOException {

  String line = value.toString();
  StringTokenizer itr = new StringTokenizer(line);
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
     output.collect(word, one);
   }
 }
}

// Reducer class that just emits the sum of the input values.

public static class Reduce extends MapReduceBase implements Reducer< Text,
IntWritable, Text, IntWritable > {
public void reduce(Text key, Iterator values,
 OutputCollector<text, intwritable=""> output,
 Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }

static int printUsage() {
System.out.println("wordcount [-m #mappers ] [-r #reducers] input_file output_file");
    ToolRunner.printGenericCommandUsage(System.out);
    return -1;
  }

public int run(String[] args) throws Exception {

    JobConf conf = new JobConf(getConf(), WordCount.class);
    conf.setJobName("wordcount");

// the keys are words (strings)
   conf.setOutputKeyClass(Text.class);
// the values are counts (ints)
   conf.setOutputValueClass(IntWritable.class);

   conf.setMapperClass(MapClass.class);
// Here we set the combiner!!!!
   conf.setCombinerClass(Reduce.class);
   conf.setReducerClass(Reduce.class);

  List other_args = new ArrayList();
   for(int i=0; i < args.length; ++i) {
     try {
       if ("-m".equals(args[i])) {
conf.setNumMapTasks(Integer.parseInt(args[++i]));
       } else if ("-r".equals(args[i])) {
conf.setNumReduceTasks(Integer.parseInt(args[++i]));
       } else {
         other_args.add(args[i]);
       }
     } catch (NumberFormatException except) {
       System.out.println("ERROR: Integer expected instead of " + args[i]);
       return printUsage();
     } catch (ArrayIndexOutOfBoundsException except) {
       System.out.println("ERROR: Required parameter missing from " +
           args[i-1]);
       return printUsage();
     }
   }
// Make sure there are exactly 2 parameters left.
  if (other_args.size() != 2) {
    System.out.println("ERROR: Wrong number of parameters: " +
        other_args.size() + " instead of 2.");
    return printUsage();
   }
   FileInputFormat.setInputPaths(conf, other_args.get(0));
   FileOutputFormat.setOutputPath(conf, new Path(other_args.get(1)));

   JobClient.runJob(conf);
   return 0;
 }

public static void main(String[] args) throws Exception {
   int res = ToolRunner.run(new Configuration(), new WordCount(), args);
   System.exit(res);
 }
}
```

COMPILE

```
$ javac WordCount.java -cp /usr/local/Cellar/hadoop
```