**Clairvoyant**

**MapReduce Exercise**

# **Documentation:**

Mapper Class: <http://hadoop.apache.org/docs/r2.4.1/api/index.html?org/apache/hadoop/mapreduce/Mapper.html>

# **Hands-On Exercise: Running a MapReduce Job**

In this exercise you will compile a MapReduce program, create a JAR, and run a MapReduce job.

**Compiling and Submitting a** **MapReduce Job**

1. Unzip the com.clairvoyant.workshop.zip file
2. Open eclipse and import the com.clairvoyant.workshop project
   1. Start up Eclipse (Shortcut to Eclipse should be on the VMs Desktop)
   2. Click File -> Import...
   3. Click the “Maven” folder and select “Existing Maven Projects” and click “Next”
   4. Under Root directory, browse for the com.clairvoyant.workshop directory that was extracted and click OK
   5. Click Finish
3. Right click on the invertedIndex project **→Run AS → Maven Build**.
4. A dialog window opens, type: **clean package** for Goals and click on **Run**. This creates a jar file under the folder target.
5. Go to the target folder from the command line.

|  |
| --- |
| cd {PATH}/com.clairvoyant.workshop.invertedindex/target |

1. Run the invertedindex MapReduce Job.

|  |
| --- |
| $ hadoop jar com.clairvoyant.workshop.invertedindex-1.0.jar com.clairvoyant.workshop.invertedindex.InvertedIndexDriver /user/cloudera/workshop/mapreduce/invertedIndex/input/ /user/cloudera/workshop/mapreduce/invertedIndex/output/ |

This should kick-off a MapReduce job for creating a file with inverted indexes.

1. View the contents of the output for your job.

|  |
| --- |
| $ hadoop fs –cat  /user/cloudera/workshop/mapreduce/invertedIndex/output/part-r-00000 | less |

1. What happens when you try to run the job again?
2. Clean up the output files produced by the previous run.

|  |
| --- |
| $ hadoop fs –rm –r /user/cloudera/workshop/mapreduce/invertedIndex/output |

**Stopping MapReduce Jobs**

It is important to note that pressing Control-C will not stop the job itself, but only kills the current process that is displaying the MapReduce jobs progress. The MapReduce job, once submitted to the Hadoop daemons, runs independently of any initiating process. In order to kill a job that’s running we need to tell the YARN ResourceManager to stop the job.

1. To list all the jobs that are running, enter:

|  |
| --- |
| $ yarn application –list |

This lists the job ids of all the running jobs.

1. The job can be killed by entering:

|  |
| --- |
| $ yarn application –kill {jobid} |

The YARN ResourceManager kills the job and the program running in the original terminal reporting its progress, informs that the job has failed.

# **Hands-On Exercise: Create a MapReduce Job**

Now that you have compiled and ran a MapReduce job, let’s have you create one yourself.

You’re going to implement the “HelloWorld” of Big Data processing projects: The Word Count Example. In this project you will count the occurrences of all the words in a document.

**Mapper Function**

1. Split each line by the ‘ ‘ (space) character. Each splitted value will be a word.
2. For each word, return the Key Value Pair (word, 1). Where the key is the word you’re processing and the value is the number 1 to identify we are counting one occurrence of that word.

Class Documentation: <http://hadoop.apache.org/docs/r2.4.1/api/index.html?org/apache/hadoop/mapreduce/Mapper.html>

**Reducer Function**

1. For each key, add up all the integer values and return a new Key Value Pair. Where the key is the word you’re processing and the value is the total after adding up the values.

Class Documentation: <http://hadoop.apache.org/docs/r2.4.1/api/index.html?org/apache/hadoop/mapreduce/Reducer.html>

Below is a diagram which illustrates how these Mapper and Reducer functions will work together:



**Completing the MapReduce Job**

1. Bring up eclipse where you had imported the com.clairvoyant.workshop project.
2. Navigate to the **com.clairvoyant.workshop.wordcount.stubs** project and implement the functions in the WordCountMapper.java and WordCountReducer.java files as per how those functions are described above.
   1. **com.clairvoyant.workshop.wordcount.solution** is available to reference if you get stuck
3. Right click on the **wordcount stubs** project **→Run AS → Maven Build**.
4. A dialog window opens, type: **clean package** for Goals and click on **Run**. This creates a jar file under the folder target.
5. Go to the target folder from the command line. Workshop-wordcount.jar
6. Go to the target folder from the command line.

|  |
| --- |
| cd {PATH}/com.clairvoyant.workshop.wordcount.stubs/target |

1. Run the wordcount MapReduce Job.

|  |
| --- |
| $ hadoop jar workshop-wordcount.jar com.clairvoyant.workshop.wordcount.WordCountDriver /user/cloudera/workshop/mapreduce/invertedIndex/input/ /user/cloudera/workshop/mapreduce/invertedIndex/word\_count\_output/ |

This should kick-off a MapReduce job for creating a file with inverted indexes.

1. View the contents of the output for your job.