CS246: Mining Massive Datasets **Winter 2014**

Problem Set 0

Due 9:30am January 14, 2014

General Instructions

This homework is to be completed individually (no collaboration is allowed). Also, you are not allowed to use any late days for the homework. This homework is worth 1% of the total course grade.

The purpose of this homework is to get you started with Hadoop. Here you will learn how to write, compile, debug and execute a simple Hadoop program. First part of the homework serves as a tutorial and the second part asks you to write your own Hadoop program.

Section 1 describes the virtual machine environment. Instead of the virtual machine, you are welcome to setup your own pseudo-distributed or fully distributed cluster if you prefer. Any version of Hadoop that is at least 1.0 will suffice. (For an easy way to set up a cluster, try Cloudera Manager: http://archive.cloudera.com/cm4/installer/latest/cloudera-manager-installer.bin.) If you choose to setup your own cluster, you are responsible for making sure the cluster is working properly. The TAs will be unable to help you debug configuration issues in your own cluster.

Section 2 explains how to use the Eclipse environment in the virtual machine, including how to create a project, how to run jobs, and how to debug jobs. Section 2.5 gives an end-to-end example of creating a project, adding code, building, running, and debugging it.

Section 3 is the actual homework assignment. There are no deliverables for sections 1 and 2. In section 3, you are asked to write and submit your own MapReduce job.

This homework requires you to upload the code and hand-in a print-out of the output for Section 3.

Regular (non-SCPD) students should submit hard copies of the answers (Section 3) either in class or in the submission box (see course website for location). For paper submission, please fill the cover sheet and submit it as a front page with your answers. You should upload your source code and any other files you used.

SCPD students should submit their answers through SCPD and also upload the code. The submission must include the answers to Section 3, the cover sheet and the usual SCPD routing form (http://scpd.stanford.edu/generalInformation/pdf/SCPD_HomeworkRouteForm.pdf).

Cover Sheet: http://cs246.stanford.edu/cover.pdf

Upload Link: http://snap.stanford.edu/submit/

Questions

1 Setting up a virtual machine

- Download and install *VirtualBox* on your machine: http://virtualbox.org/wiki/Downloads
- Download the *Cloudera Quickstart VM* at http://www.cloudera.com/content/dev-center/en/home/developer-admin-resources/quickstart-vm.html
- Uncompress the VM archive. It is compressed with 7-Zip. If needed, you can download a tool to uncompress the archive at http://www.7-zip.org/.
- Start *VirtualBox* and click *Import Appliance*. Click the folder icon beside the location field. Browse to the uncompressed archive folder, select the .ovf file, and click the *Open* button. Click the *Continue* button. Click the *Import* button.
- Your virtual machine should now appear in the left column. Select it and click on *Start* to launch it. Username and password are "cloudera" and "cloud era".
- *Optional:* Open the network properties for the virtual machine. Click on the *Adapter 2* tab. Enable the adapter and select *Host-only Adapter*. If you do this step, you will be able to connect to the running virtual machine from the host OS at 192.168.56.101.

Virtual machine includes the following software

- CentOS 6.2
- JDK 6 (1.6.0_32)
- Hadoop 2.0.0
- Eclipse 4.2.6 (Juno)

The login user is `cloudera`, and the password for that account is `cloudera`.

2 Running Hadoop jobs

Generally Hadoop can be run in three modes.

1. **Standalone (or local) mode:** There are no daemons used in this mode. Hadoop uses the local file system as an substitute for HDFS file system. The jobs will run as if there is 1 mapper and 1 reducer.

2. **Pseudo-distributed mode:** All the daemons run on a single machine and this setting mimics the behavior of a cluster. All the daemons run on your machine locally using the HDFS protocol. There can be multiple mappers and reducers.

3. **Fully-distributed mode:** This is how Hadoop runs on a real cluster.

In this homework we will show you how to run Hadoop jobs in Standalone mode (very useful for developing and debugging) and also in Pseudo-distributed mode (to mimic the behavior of a cluster environment).

2.1 Creating a Hadoop project in Eclipse

(There is a plugin for Eclipse that makes it simple to create a new Hadoop project and execute Hadoop jobs, but the plugin is only well maintained for Hadoop 1.0.4, which is a rather old version of Hadoop. There is a project at https://github.com/winghc/hadoop2x-eclipse-plugin that is working to update the plugin for Hadoop 2.0. You can try it out if you like, but your milage may vary.)

To create a project:

1. Open or create the `~/.m2/settings.xml` file and make sure it has the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
  <profiles>
    <profile>
      <id>standard-extra-repos</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <repositories>
        <repository>
          <id>central</id>
          <url>http://repo.maven.apache.org/maven2/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
        <repository>
          <id>cloudera</id>
          <url>
```

```
            https://repository.cloudera.com/artifactory/cloudera-repos
          </url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
    </profile>
  </profiles>
</settings>
```

2. Open Eclipse and select *File* → *New* → *Project...*.
3. Expand the *Maven* node, select *Maven Project*, and click the *Next* > button.
4. On the next screen, click the *Next* > button.
5. On the next screen, when the archetypes have loaded, select *maven-archetype-quickstart* and click the *Next* > button.
6. On the next screen, enter a group name in the *Group Id* field, and enter a project name in the *Artifact Id*. Click the *Finish* button.
7. In the package explorer, expand the project node and double-click the `pom.xml` file to open it.
8. Replace the current "dependencies" section with the following content:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>jdk.tools</groupId>
            <artifactId>jdk.tools</artifactId>
            <version>1.6</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>2.0.0-cdh4.0.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-auth</artifactId>
            <version>2.0.0-cdh4.0.0</version>
```

```
            </dependency>
            <dependency>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>hadoop-common</artifactId>
                <version>2.0.0-cdh4.0.0</version>
            </dependency>
            <dependency>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>hadoop-core</artifactId>
                <version>2.0.0-mr1-cdh4.0.1</version>
            </dependency>
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit-dep</artifactId>
                <version>4.8.2</version>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-auth</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-core</artifactId>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
```

```
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.1</version>
                <configuration>
                    <source>1.6</source>
                    <target>1.6</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

9. Save the file.
10. Right-click on the project node and select *Maven* → *Update Project*.

You can now create classes in the `src` directory. After writing your code, build the JAR file by right-clicking on the project node and selecting *Run As* → *Maven install*.

2.2 Running Hadoop jobs in standalone mode

After creating a project, adding source code, and building the JAR file as outlined above, the JAR file will be located at `/workspace/`$< project >$`/target` directory.

Open a terminal and run the following command:

`hadoop jar ~/workspace/`$< project >$`/target/`$< project >$`-0.0.1-SNAPSHOT.jar \`

`-D mapped.task.tracker=local -D fs.defaultFS=local` $< args >$

You will see all of the output from the map and reduce tasks in the terminal.

2.3 Running Hadoop jobs in pseudo-distributed mode

Open a terminal and run the following command:

`hadoop jar ~/workspace/`$< project >$`/target/`$< project >$`-0.0.1-SNAPSHOT.jar` $< args >$

To see all running jobs, run the following command:

`hadoop job -list`

To kill a running job, find the job's ID and then run the following command:

`hadoop job -kill` $< id >$

2.4 Debugging Hadoop jobs

To debug an issue with a job, the easiest approach is to add print statements into the source file and run the job in standalone mode. The print statements will appear in the terminal output. When running your job in pseudo-distributed mode, the output from the job is logged in the task tracker's log files, which can be accessed most easily by pointing a web browser to port 50030 of the server. From the job tracker web page, you can drill down into the failing job, the failing task, the failed attempt, and finally the log files. Note that the logs for stdout and stderr are separated, which can be useful when trying to isolate specific debugging print statements.

If you enabled the second network adapter in the VM setup, you can point your local browser to `http://192.168.56.101:50030/` to access the job tracker page. Note, though, that when you follow links that lead to the task tracker web page, the links point to `localhost.locadomain`, which means your browser will return a page not found error. Simply replace `localhost.locadomain` with `192.168.56.101` in the URL bar and press enter to load the correct page.

2.5 Example project

In this section you will create a new Eclipse Hadoop project, compile, and execute it. The program will count the frequency of all the words in a given large text file. In your virtual machine, Hadoop, Java environment and Eclipse have already been pre-installed.

- Edit the `~/.m2/settings.xml` file as outlined above. See Figure 1

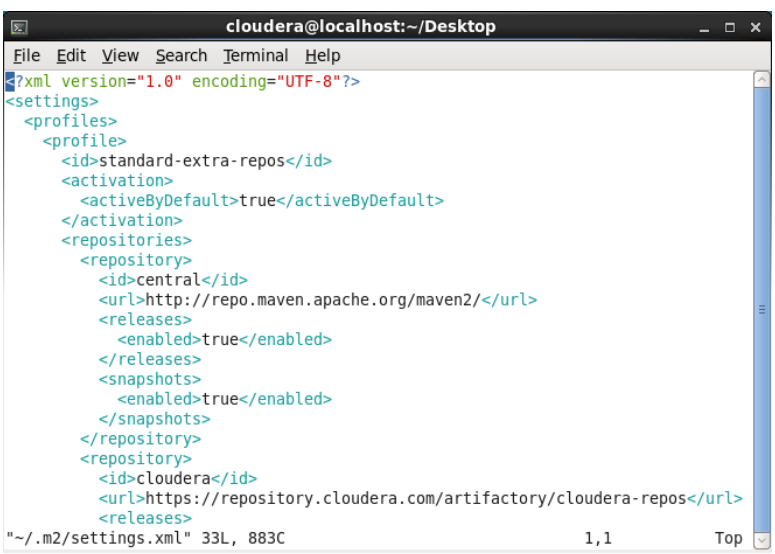

Figure 1: Create a Hadoop Project.

- Open Eclipse and create a new project as outlined above. See Figures 2-9.

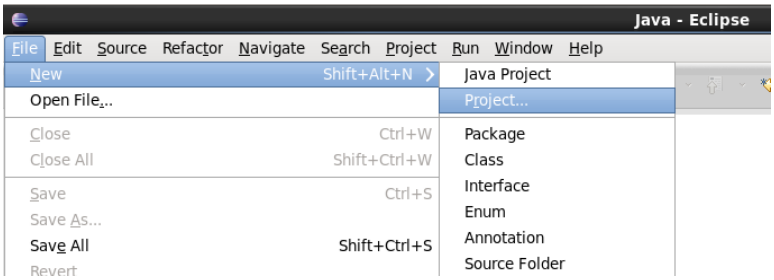

Figure 2: Create a Hadoop Project.

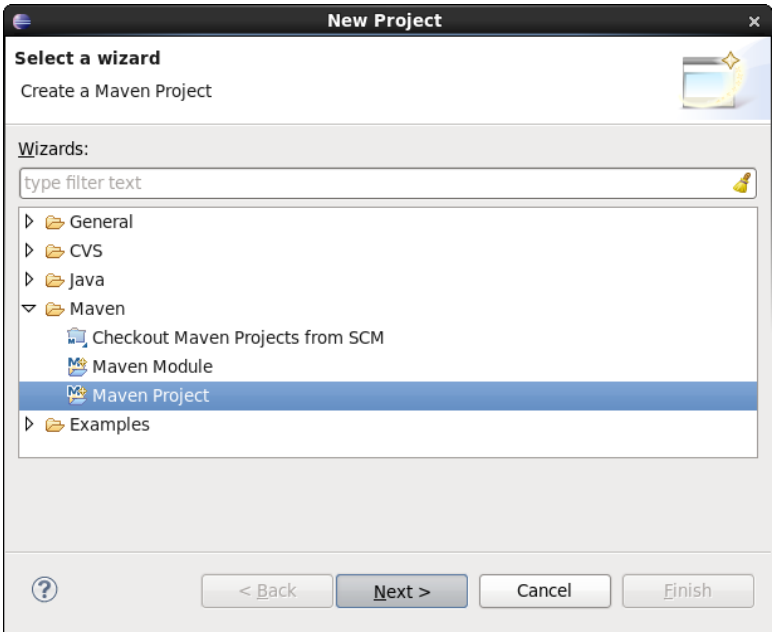

Figure 3: Create a Hadoop Project.

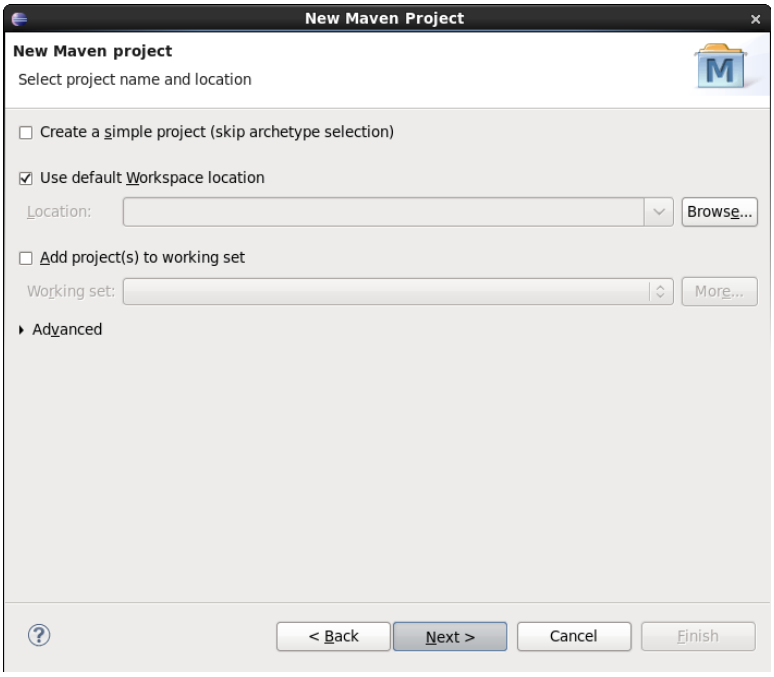

Figure 4: Create a Hadoop Project.

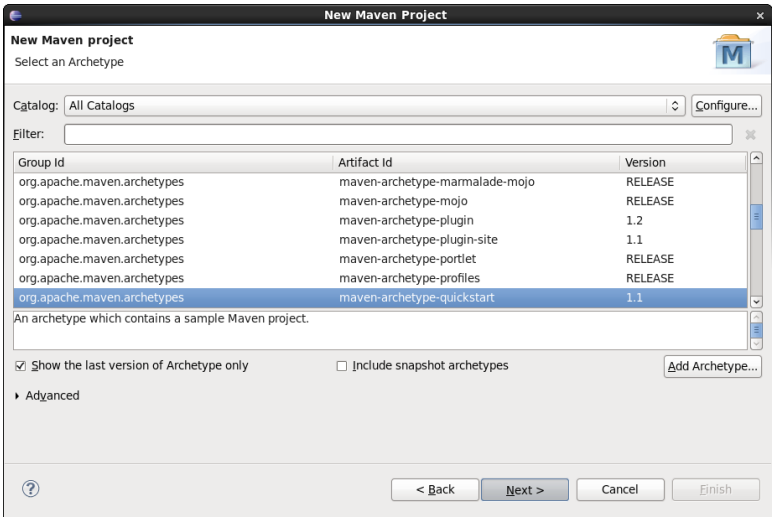

Figure 5: Create a Hadoop Project.

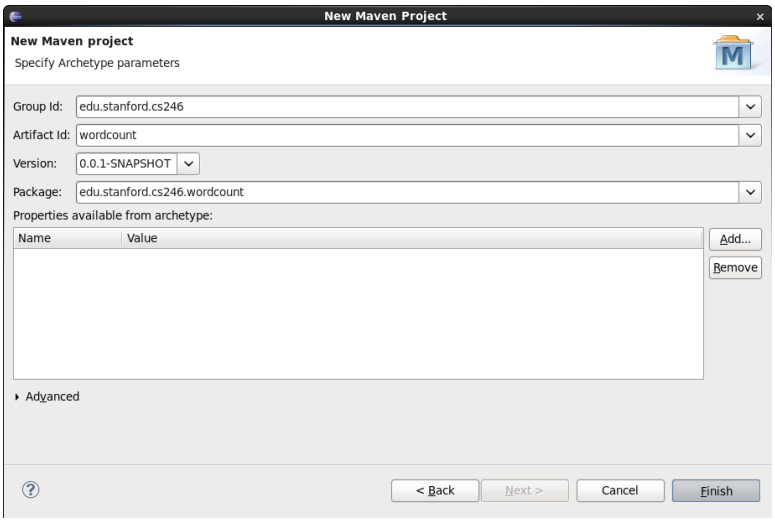

Figure 6: Create a Hadoop Project.

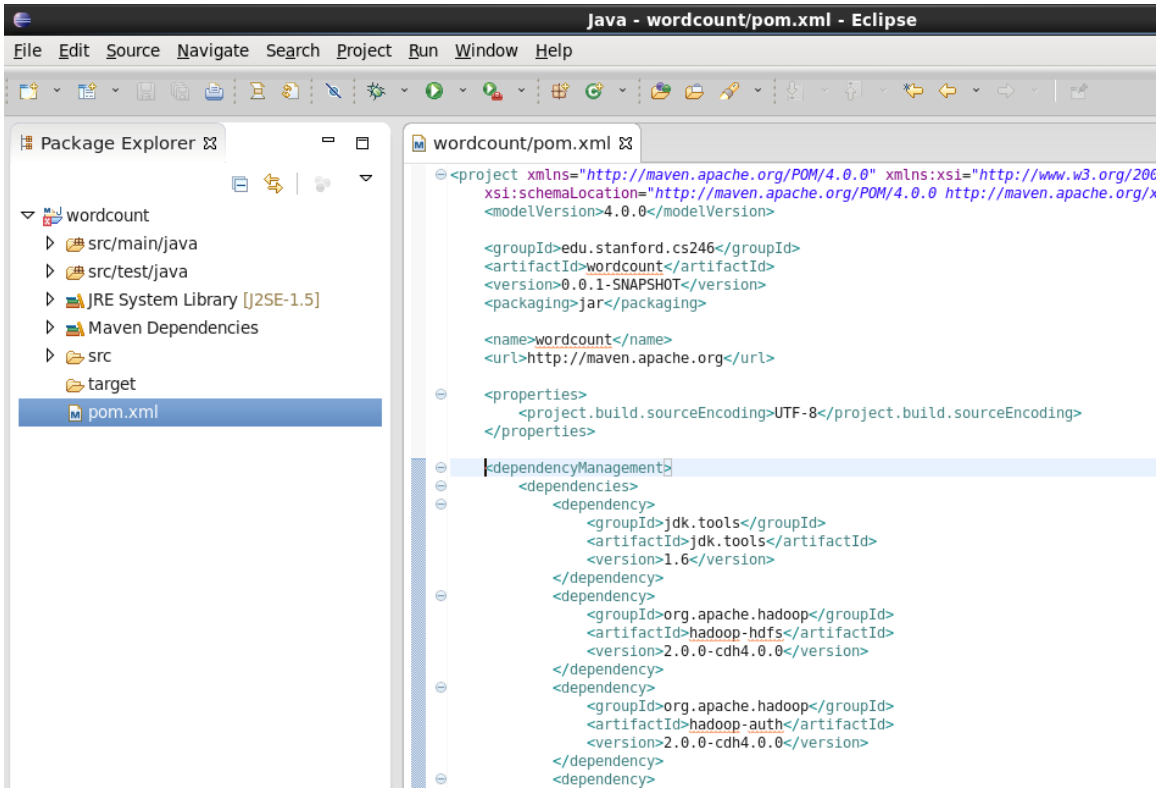

Figure 7: Create a Hadoop Project.

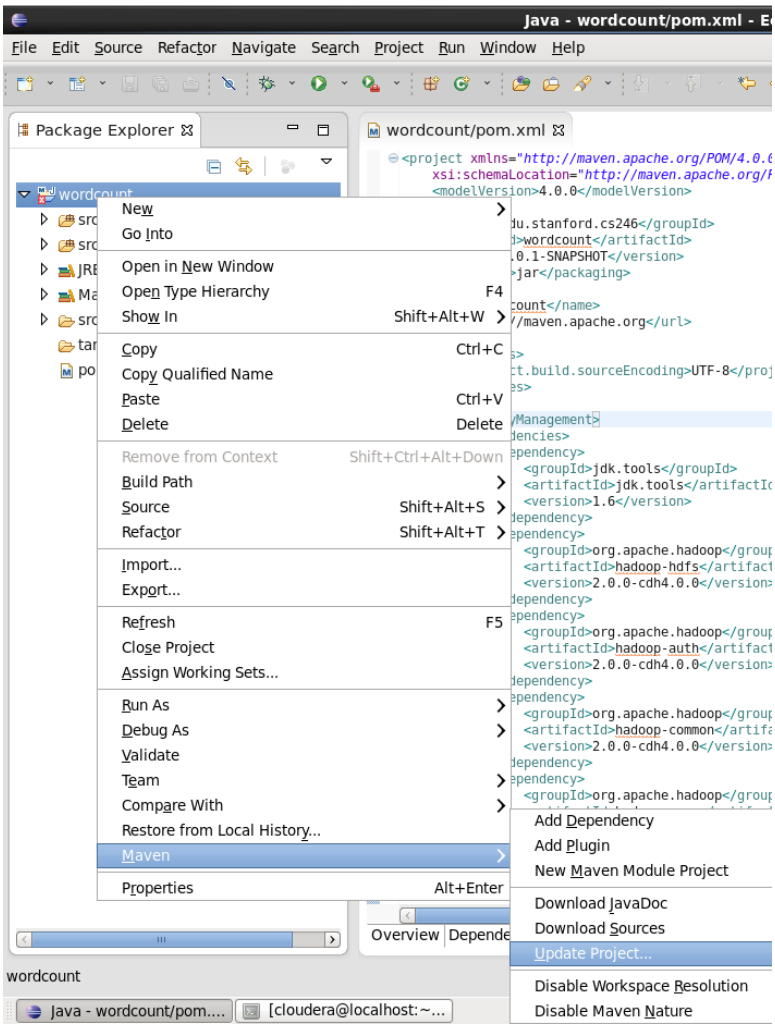

Figure 8: Create a Hadoop Project.

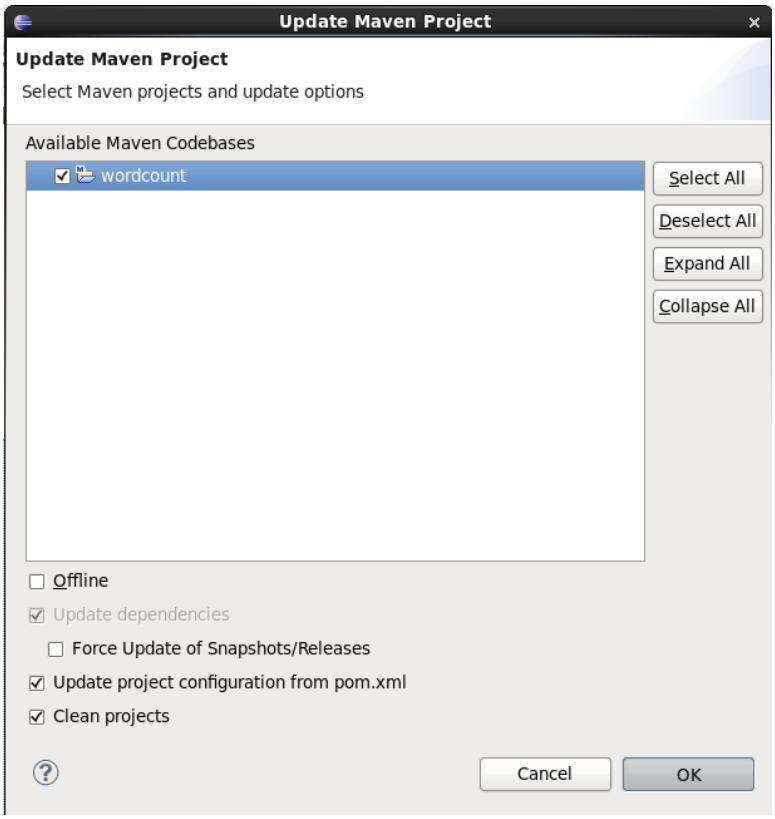

Figure 9: Create a Hadoop Project.

- The project will contain a stub source file in the `src/main/java` directory that we will not use. Instead, create a new class called WordCount. From the *File* menu, select *New* → *Class*. See Figure 10

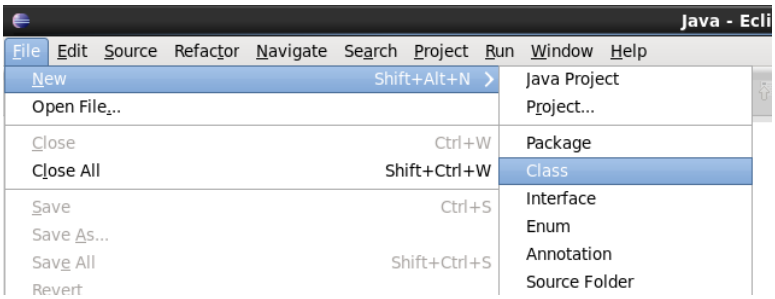

Figure 10: Create java file.

- On the next screen, enter the package name (e.g, the group ID plus the project name) in the *Package* field. Enter `WordCount` as the *Name*. See Figure 11.

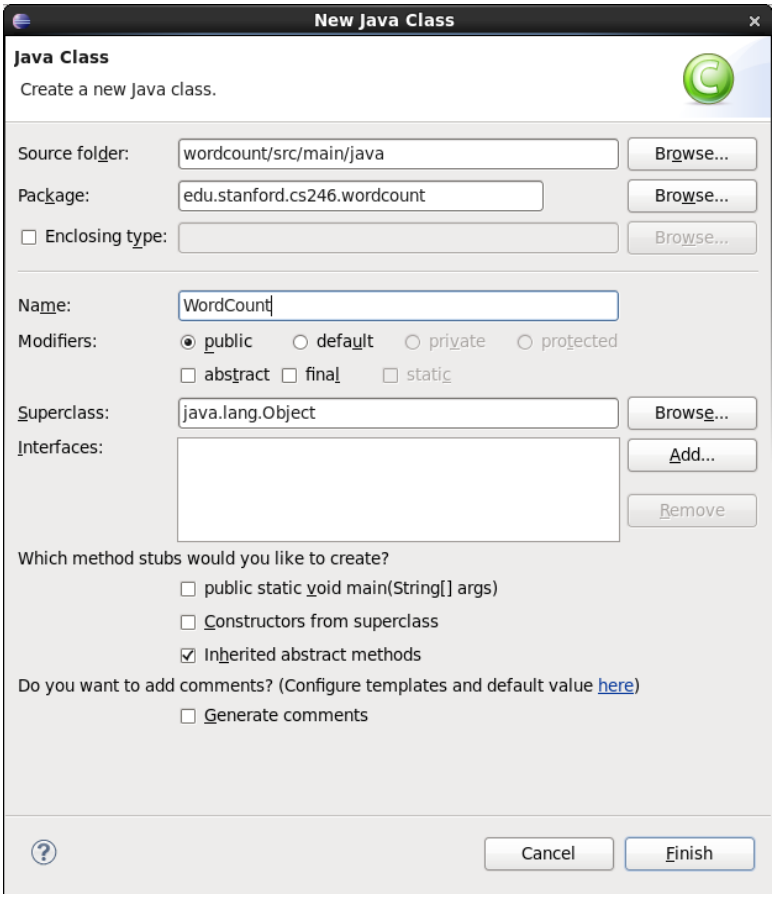

Figure 11: Create java file.

- In the *Superclass* field, enter `Configured` and click the *Browse* button. From the pop-up window select ***Configured*** — *org.apache.hadoop.conf* and click the *OK* button. See Figure 12.

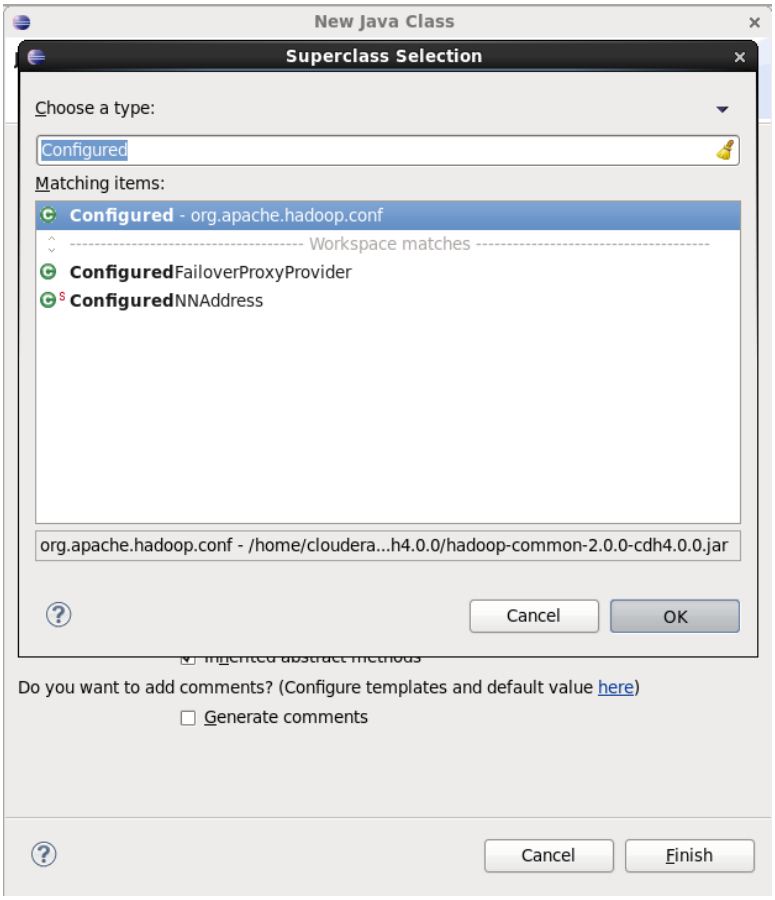

Figure 12: Create java file.

- In the *Interfaces* section, click the *Add* button. From the pop-up window select ***Tool*** — *org.apache.hadoop.util* and click the *OK* button. See Figure 13.

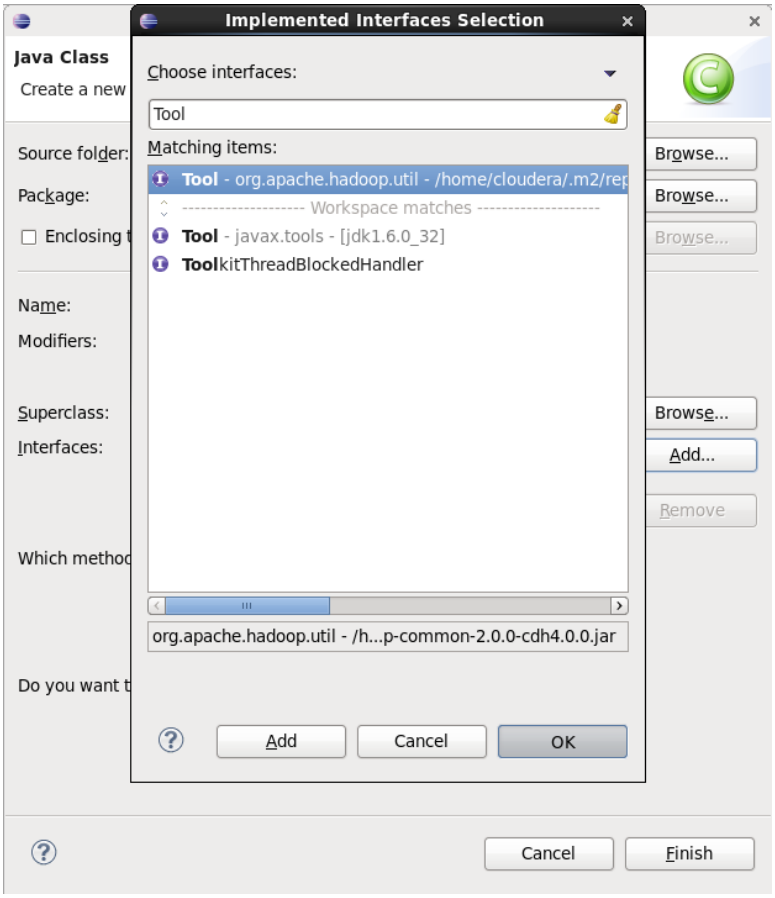

Figure 13: Create java file.

- Check the boxes for *public static void main(String args[])* and *Inherited abstract methods* and click the *Finish* button. Se Figure 14

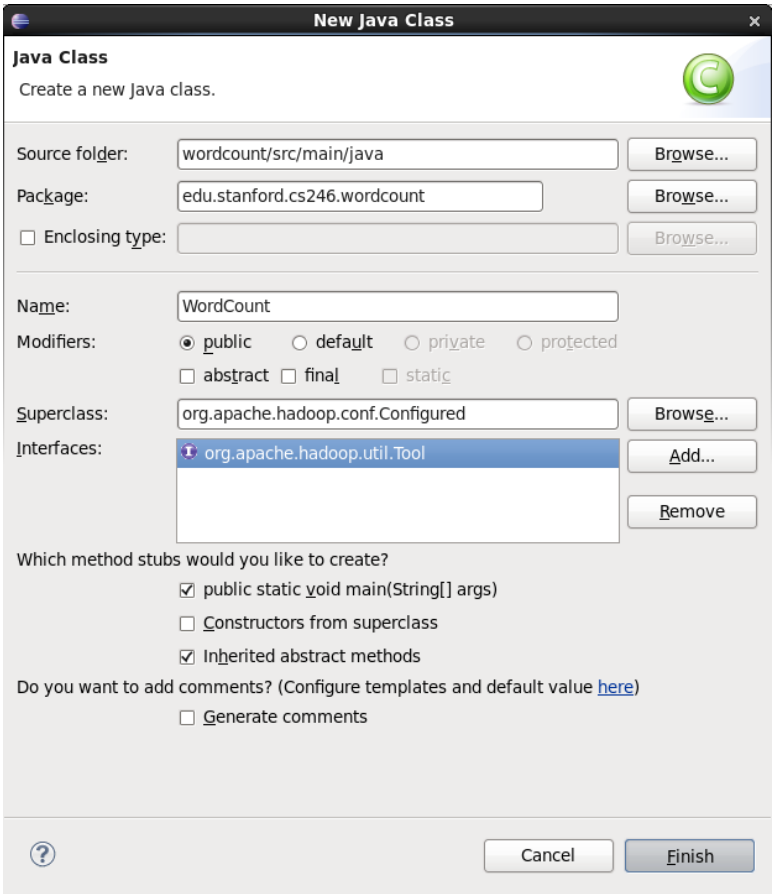

Figure 14: Create WordCount.java.

- You will now have a rough skeleton of a Java file as in Figure 15. You can now add code to this class to implement your Hadoop job.

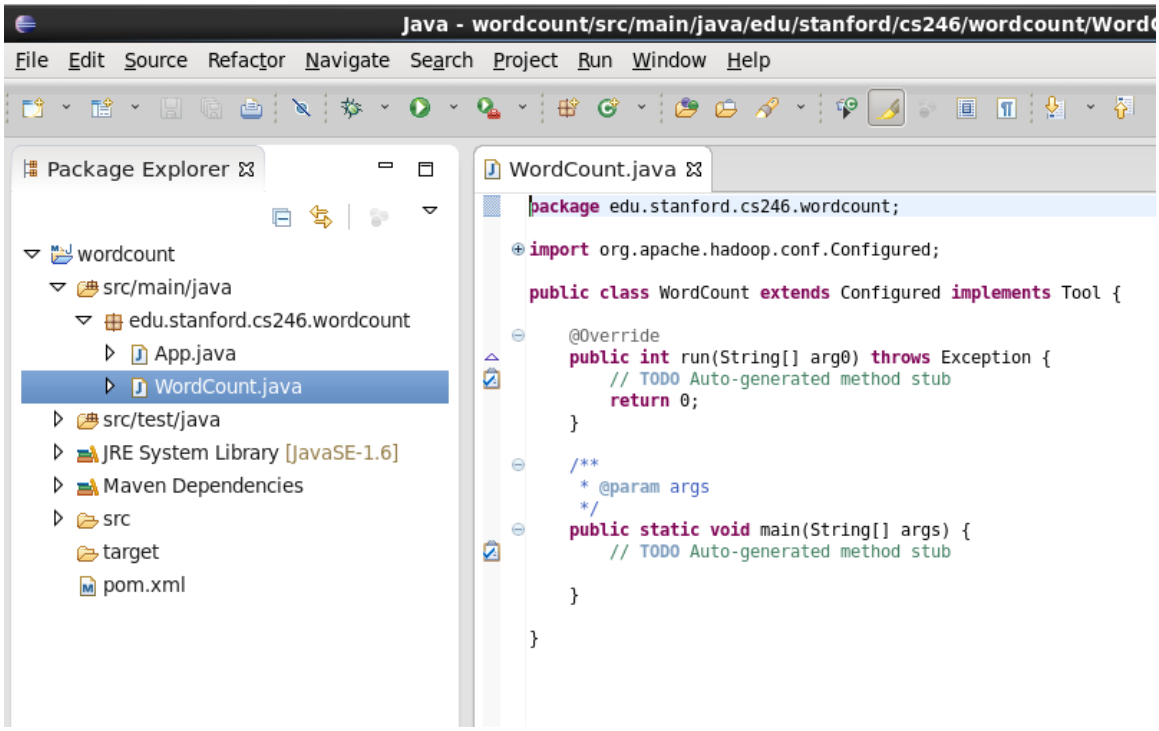

Figure 15: Create WordCount.java.

- Rather than implement a job from scratch, copy the contents from http://snap.stanford.edu/class/cs246-data-2014/WordCount.java and paste it into the `WordCount.java` file. Be careful to leave the package statement at the top intact. See Figure 16. The code in `WordCount.java` calculates the frequency of each word in a given dataset.

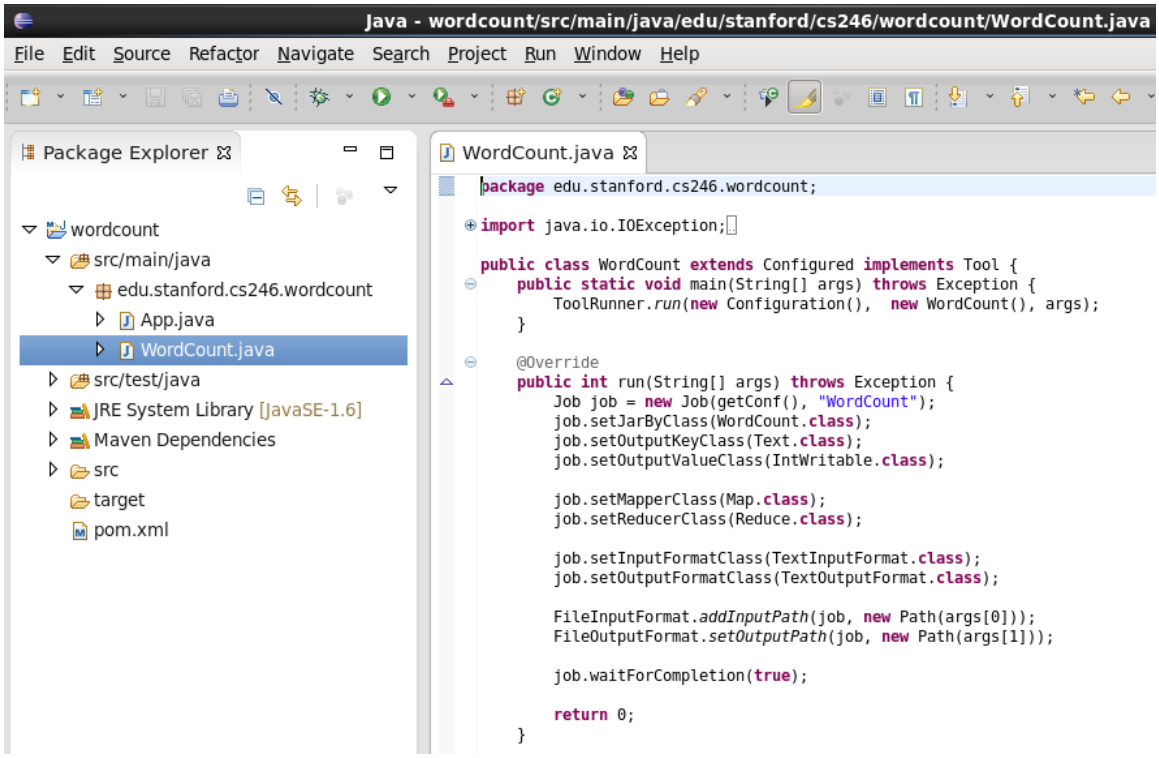

Figure 16: Create WordCount.java.

- Build the project by right-clicking the project node and selecting *Run As* → *Maven install*. See Figure 17.

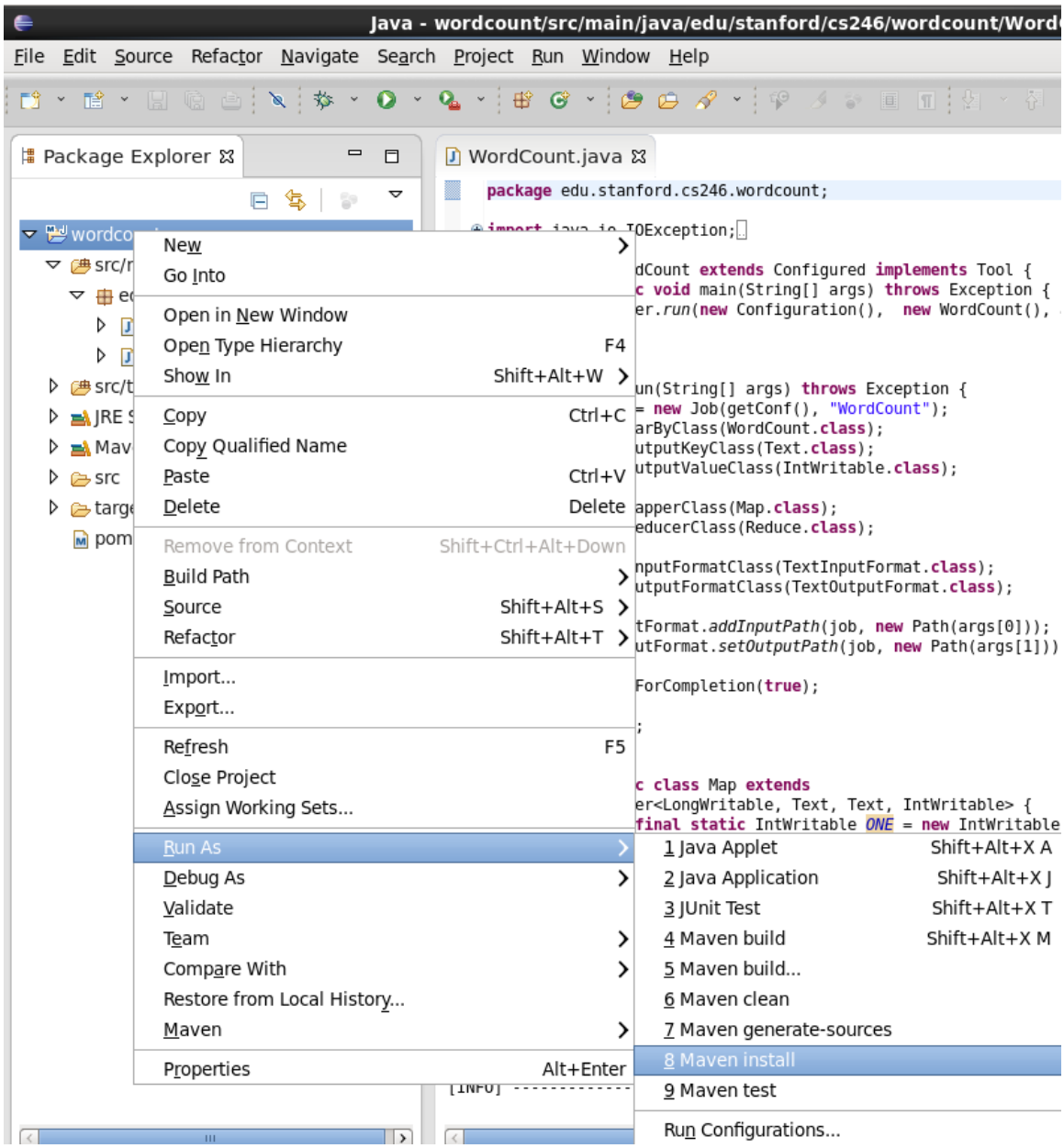

Figure 17: Create WordCount.java.

- Download the *Complete Works of William Shakespeare* from Project Gutenberg at http://www.gutenberg.org/cache/epub/100/pg100.txt.
- Open a terminal and change to the directory where the dataset was stored.
- Run the command:

```
hadoop jar ~/workspace/wordcount/target/wordcount-0.0.1-SNAPSHOT.jar \
edu.stanford.cs246.wordcount.WordCount -D mapred.job.tracker=local \
-D fs.defaultFS=local dataset output
```

See Figure 18

```
[cloudera@localhost Desktop]$ wget http://www.gutenberg.org/cache/epub/100/pg100.txt
--2014-01-01 16:16:00--  http://www.gutenberg.org/cache/epub/100/pg100.txt
Resolving www.gutenberg.org... 152.19.134.47
Connecting to www.gutenberg.org|152.19.134.47|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5589890 (5.3M) [text/plain]
Saving to: “pg100.txt”

100%[===================================================================================>] 5,589,890   1.24M/s   in 5.8s

2014-01-01 16:16:06 (942 KB/s) - “pg100.txt” saved [5589890/5589890]

[cloudera@localhost Desktop]$ hadoop jar ~/workspace/wordcount/target/wordcount-0.0.1-SNAPSHOT.jar edu.stanford.cs246.wordcount.W
ordCount -D mapred.job.tracker=local -D fs.defaultFS=local pg100.txt output
14/01/01 16:16:35 WARN fs.FileSystem: "local" is a deprecated filesystem name. Use "file:///" instead.
14/01/01 16:16:35 WARN fs.FileSystem: "local" is a deprecated filesystem name. Use "file:///" instead.
14/01/01 16:16:35 WARN fs.FileSystem: "local" is a deprecated filesystem name. Use "file:///" instead.
14/01/01 16:16:35 WARN conf.Configuration: session.id is deprecated. Instead, use dfs.metrics.session-id
14/01/01 16:16:35 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
14/01/01 16:16:36 WARN fs.FileSystem: "local" is a deprecated filesystem name. Use "file:///" instead.
14/01/01 16:16:36 WARN fs.FileSystem: "local" is a deprecated filesystem name. Use "file:///" instead.
14/01/01 16:16:36 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool f
or the same.
```

Figure 18: Run WordCount job.

- If the job succeeds, you will see an `output` directory in the current directory that contains a file called `part-00000`. The `part-00000` file contains the output from the job. See Figure 19

```
[cloudera@localhost Desktop]$ head output/part-r-00000
        119383
"       241
"'Tis   1
"A      4
"AS-IS".        1
"Air,"  1
"Alas,  1
"Amen"  2
"Amen"? 1
"Amen," 1
```

Figure 19: Run WordCount job.

- Run the command:

  `hadoop fs -ls`

  The command will list the contents of your home directory in HDFS, which should be empty, resulting in no output.

- Run the command:

  `hadoop fs -copyFromLocal pg100.txt`

  to copy the dataset folder into HDFS.

- Run the command:

  `hadoop fs -ls`

again. You should see the `dataset` directory listed, as in Figure 20 indicating that the dataset is in HDFS.

```
[cloudera@localhost Desktop]$ hadoop fs -ls
[cloudera@localhost Desktop]$ hadoop fs -copyFromLocal pg100.txt
[cloudera@localhost Desktop]$ hadoop fs -ls
Found 1 items
-rw-r--r--   3 cloudera cloudera    5589890 2014-01-01 16:19 pg100.txt
[cloudera@localhost Desktop]$
```

Figure 20: Run WordCount job.

- Run the command:

  `hadoop jar ~/workspace/WordCount/target/WordCount-0.0.1-SNAPSHOT.jar \`

  `edu.stanford.cs246.wordcount.WordCount pg100.txt output`

  See Figure 21. If the job fails, you will see a message indicating that the job failed. Otherwise, you can assume the job succeeded.

```
[cloudera@localhost Desktop]$ hadoop jar ~/workspace/wordcount/target/wordcount-0.0.1-SNAPSHOT.jar edu.stanford.cs246.wordcount.W
ordCount pg100.txt output
14/01/01 16:22:08 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool f
or the same.
14/01/01 16:22:09 INFO input.FileInputFormat: Total input paths to process : 1
14/01/01 16:22:09 INFO mapred.JobClient: Running job: job_201401011523_0001
14/01/01 16:22:11 INFO mapred.JobClient:  map 0% reduce 0%
14/01/01 16:22:24 INFO mapred.JobClient:  map 100% reduce 0%
14/01/01 16:22:32 INFO mapred.JobClient:  map 100% reduce 100%
14/01/01 16:22:34 INFO mapred.JobClient: Job complete: job_201401011523_0001
14/01/01 16:22:34 INFO mapred.JobClient: Counters: 32
14/01/01 16:22:34 INFO mapred.JobClient:   File System Counters
14/01/01 16:22:34 INFO mapred.JobClient:     FILE: Number of bytes read=2491864
14/01/01 16:22:34 INFO mapred.JobClient:     FILE: Number of bytes written=3797929
14/01/01 16:22:34 INFO mapred.JobClient:     FILE: Number of read operations=0
14/01/01 16:22:34 INFO mapred.JobClient:     FILE: Number of large read operations=0
14/01/01 16:22:34 INFO mapred.JobClient:     FILE: Number of write operations=0
```

Figure 21: Run WordCount job.

- Run the command:

  `hadoop fs -ls output`

  You should see an output file for each reducer. Since there was only one reducer for this job, you should only see one `part-*` file. Note that sometimes the files will be called `part-NNNNN`, and sometimes they'll be called `part-r-NNNNN`. See Figure 22

```
[cloudera@localhost Desktop]$ hadoop fs -ls output
Found 3 items
-rw-r--r--   3 cloudera cloudera          0 2014-01-01 16:22 output/_SUCCESS
drwxr-xr-x   - cloudera cloudera          0 2014-01-01 16:22 output/_logs
-rw-r--r--   3 cloudera cloudera     720989 2014-01-01 16:22 output/part-r-00000
```

Figure 22: Run WordCount job.

- Run the command:

  `hadoop fs -cat output/part\* | head`

  You should see the same output as when you ran the job locally, as shown in Figure 23

```
[cloudera@localhost Desktop]$ hadoop fs -cat output/part\* | head
        119383
"       241
"'Tis   1
"A      4
"AS-IS".        1
"Air,"  1
"Alas,  1
"Amen"  2
"Amen"? 1
"Amen," 1
cat: Unable to write to output stream.
```

Figure 23: Run WordCount job.

- To view the job's logs, open the browser in the VM and point it to `http://localhost:50030` as in Figure 24

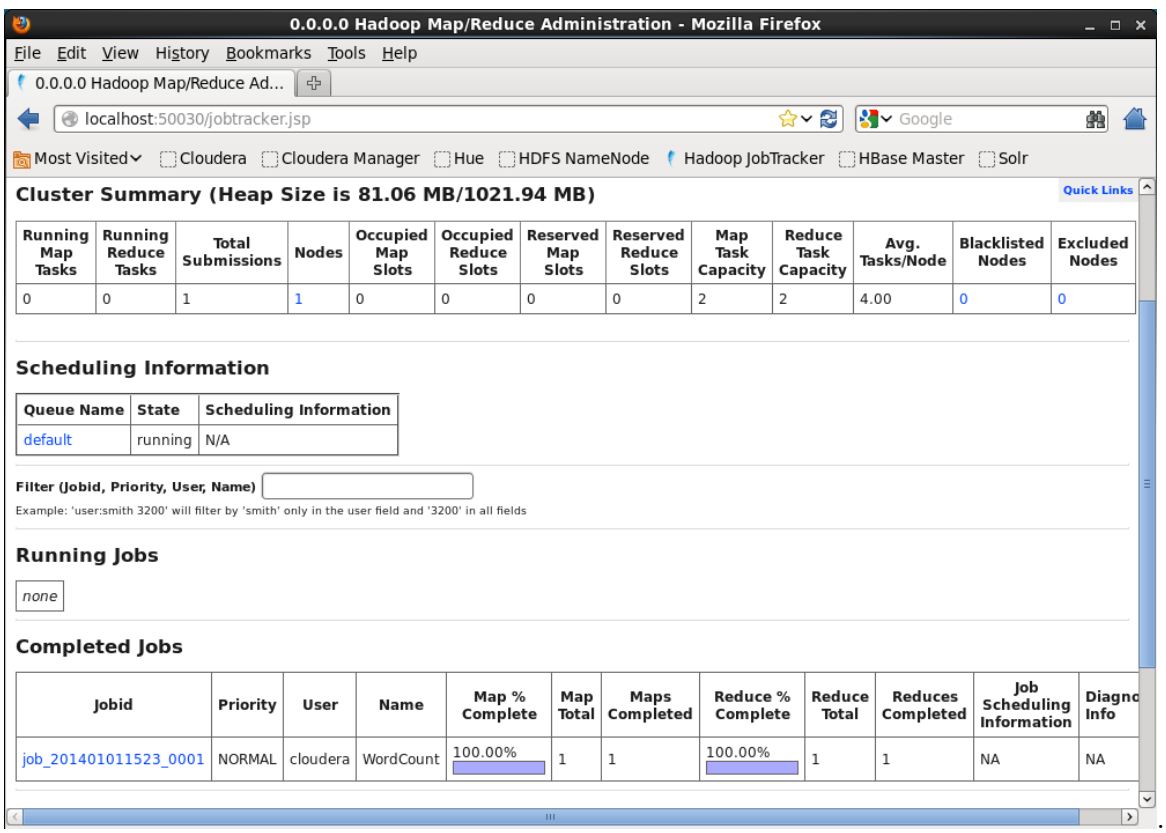

Figure 24: View WordCount job logs.

- Click on the link for the completed job. See Figure 25.

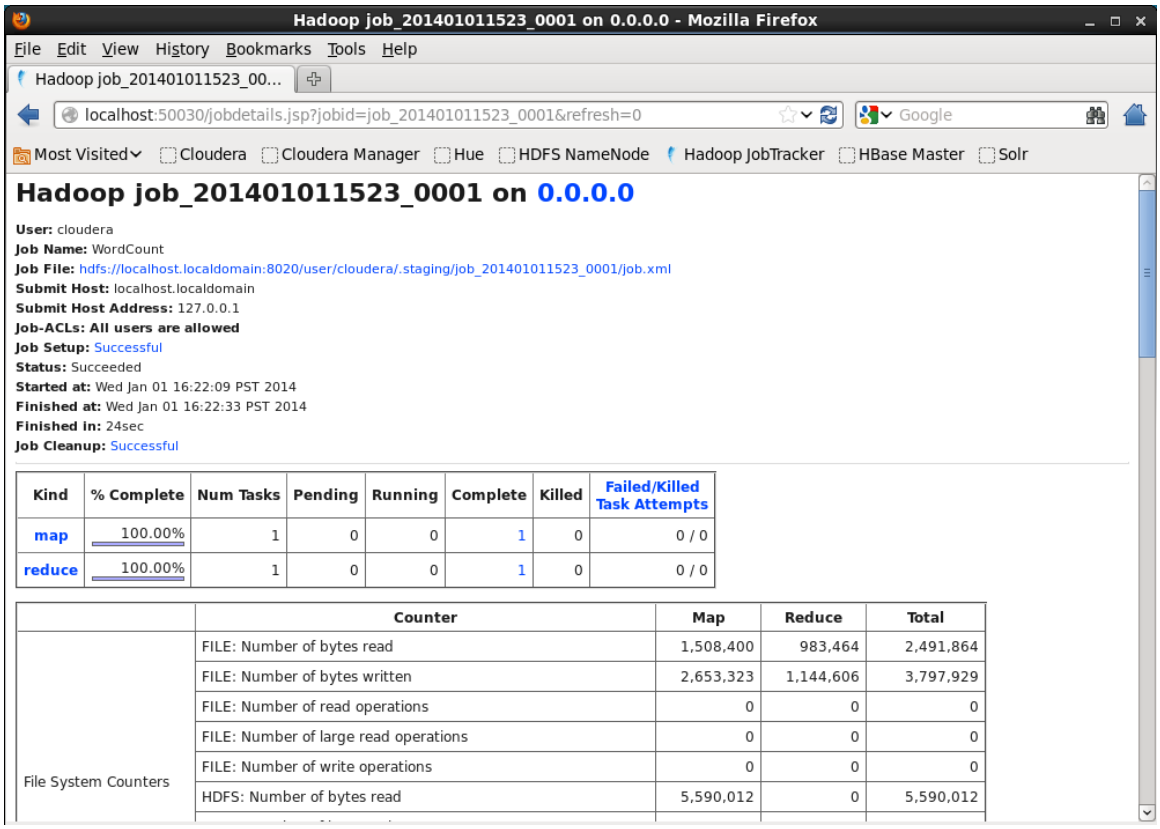

Figure 25: View WordCount job logs.

- Click the link for the map tasks. See Figure 26.

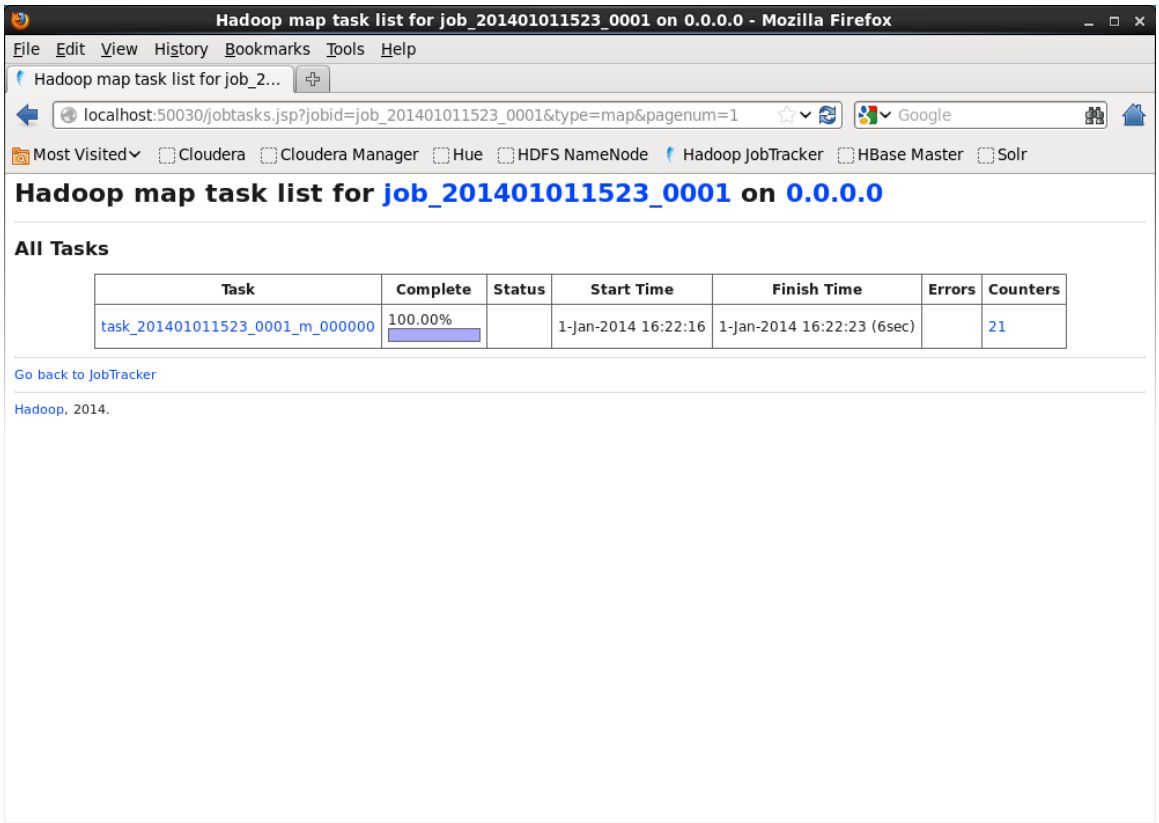

Figure 26: View WordCount job logs.

- Click the link for the first attempt. See Figure 27.

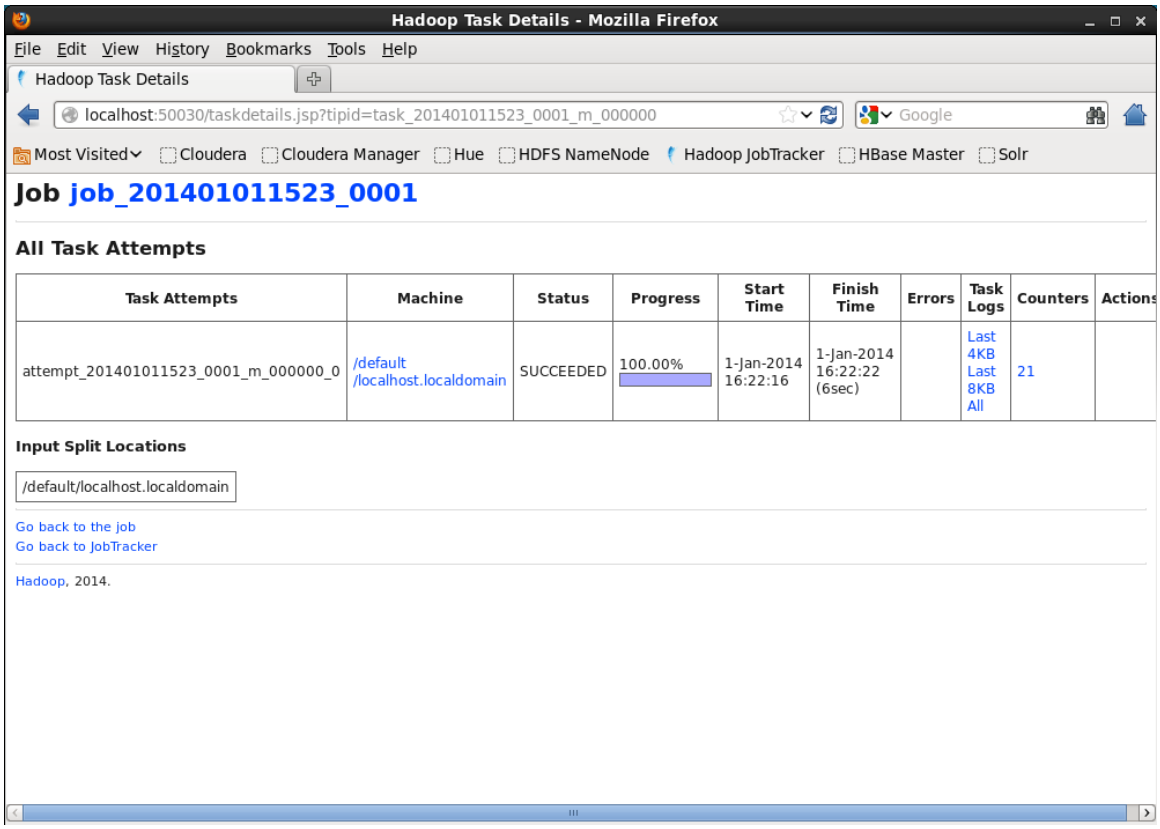

Figure 27: View WordCount job logs.

- Click the link for the full logs. See Figure 28.

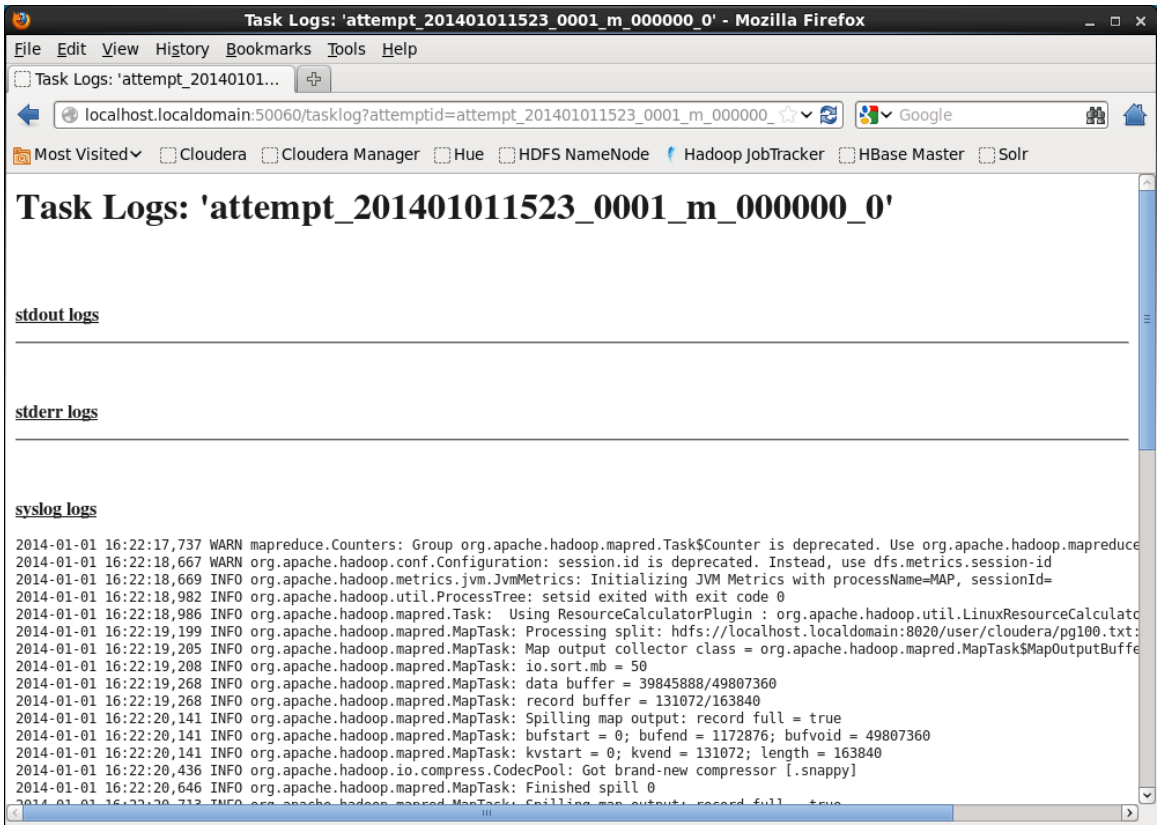

Figure 28: View WordCount job logs.

2.6 Using your local machine for development

If you enabled the second network adapter, you can use your own local machine for development, including your local IDE. If order to do that, you'll need to install a copy of Hadoop locally. The easiest way to do that is to simply download the archive from http://archive.cloudera.com/cdh4/cdh/4/hadoop-2.0.0-cdh4.4.0.tar.gz and unpack it. In the unpacked archive, you'll find a `etc/hadoop-mapreduce1` directory. In that directory, open the `core-site.xml` file and modify it as follows:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://192.168.56.101:8020</value>
    </property>
</configuration>
```

Next, open the `mapred-site.xml` file in the same directory and modify it as follows:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
   <property>
      <name>mapred.job.tracker</name>
      <value>192.168.56.101:8021</value>
   </property>
</configuration>
```

After making those modifications, update your command path to include the `bin-mapreduce1` directory and set the `HADOOP_CONF_DIR` environment variable to be the path to the `etc/hadoop-mapreduce1` directory. You should now be able to execute Hadoop commands from your local terminal just as you would from the terminal in the virtual machine. You may also want to set the `HADOOP_USER_NAME` environment variable to `cloudera` to let you masquerade as the *cloudera* user. When you use the VM directly, you're running as the *cloudera* user.

Further Hadoop tutorials

- Yahoo! Hadoop Tutorial: http://developer.yahoo.com/hadoop/tutorial/
- Cloudera Hadoop Tutorial: http://www.cloudera.com/content/cloudera-content/cloudera-docs/HadoopTutorial/CDH4/Hadoop-Tutorial.html
- How to Debug MapReduce Programs: http://wiki.apache.org/hadoop/HowToDebugMapReducePrograms

Further Eclipse tutorials

- Genera Eclipse tutorial: http://www.vogella.com/articles/Eclipse/article.html.
- Tutorial on how to use the Eclipse debugger: http://www.vogella.com/articles/EclipseDebugging/article.html.

3 Task: Write your own Hadoop Job

Now you will write your first MapReduce job to accomplish the following task:

- Write a Hadoop MapReduce program which outputs the number of words that start with each letter. This means that for every letter we want to count the total number of words that start with that letter. In your implementation ignore the letter case, *i.e.*, consider all words as lower case. You can ignore all non-alphabetic characters.
- Run your program over the same input data as above.

What to hand-in: Hand-in the printout of the output file and upload the source code.