IBM Software
Information Management

Lab 01: HDFS, MapReduce, Pig, Hive, and Jaql

Hands-On Lab

Table of Contents

1 Introduction

The overwhelming trend towards digital services, combined with cheap storage, has generated massive amounts of data that enterprises need to effectively gather, process, and analyze. Techniques from the data warehousing and high-performance computing communities are invaluable for many enterprises. However, often times their cost or complexity of scale-up discourages the accumulation of data without an immediate need. As valuable knowledge may nevertheless be buried in this data, related scaled-up technologies have been developed. Examples include Google's MapReduce, and the open-source implementation, Apache Hadoop.

Hadoop is an open-source project administered by the Apache Software Foundation. Hadoop's contributors work for some of the world's biggest technology companies. That diverse, motivated community has produced a collaborative platform for consolidating, combining and understanding data.

Technically, Hadoop consists of two key services: data storage using the Hadoop Distributed File System (HDFS) and large scale parallel data processing using a technique called MapReduce

2 About this Lab

After completing this hands-on lab, you'll be able to:

- Use Hadoop commands to explore the HDFS on the Hadoop system
- Use Hadoop commands to run a sample MapReduce program on the Hadoop system
- Explore Pig, Hive and Jaql

3 Environment Setup Requirements

To complete this lab you will need the following:

1. InfoSphere BigInsights Bootcamp VMware® image
2. VMware Player 2.x or VMware Workstation 5.x or later

For help on how to obtain these components please follow the instructions specified in VMware Basics and Introduction from module 1.

3.1 Getting Started

To prepare for the contents of this lab, you must go through the process of getting all of the Hadoop components started.

1. Start the VMware image by clicking the 

button in VMware Workstation if it is not already on.
2. Log in to the VMware virtual machine using the following information:
   - User: biadmin
   - Password: password

3. Open Gnome Command Prompt Window by right-clicking on the Desktop and selecting “Open in Terminal”.

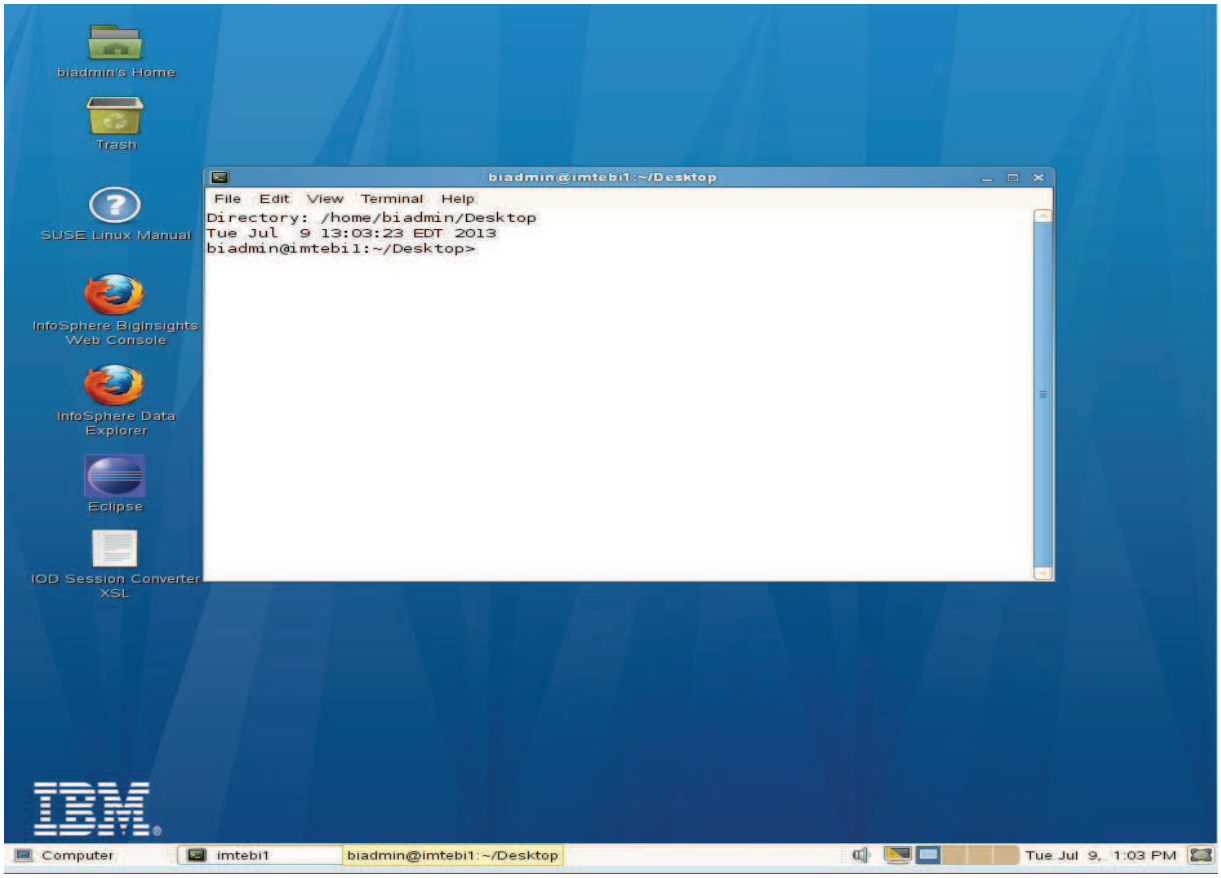


Figure 1 - Open a new terminal window

4. Change to the **$BIGINSIGHTS_HOME** (which by default is set to /opt/ibm/biginsights).

```
cd $BIGINSIGHTS_HOME/bin
```

or

```
cd /opt/ibm/biginsights/bin
```

5. Start the Hadoop components (daemons) on the BigInsights server. You can practice starting all components with these commands. Please note they will take a few minutes to run:

```
./start-all.sh
```

The following figure shows the different Hadoop components starting.

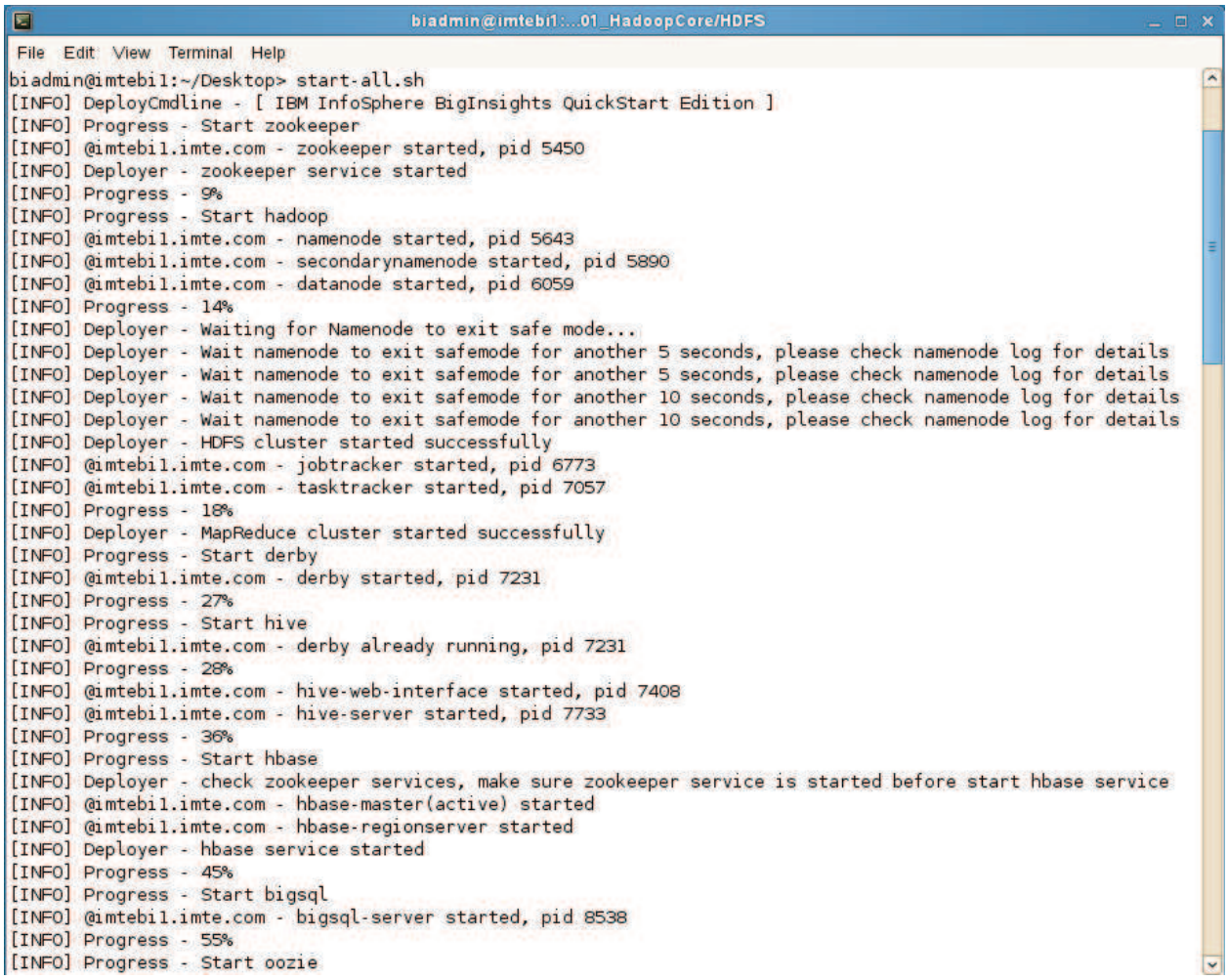


Figure 2 - Starting Hadoop components

Note: You may get an error that the server has not started, please be patient as it does take some time for the server to complete start.

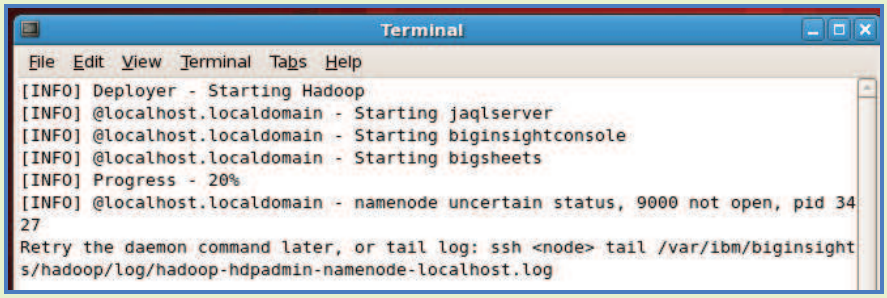


Figure 3 - Hadoop component error

6. Sometimes certain hadoop components may fail to start. You can start and stop the failed components one at a time by using **start.sh** or **stop.sh** respectively. For example, to start and stop Hadoop use:

```
./start.sh hadoop
./stop.sh hadoop
```

In the following example, the console component failed. The particular component was then started again using the **`./start.sh console`** command. It then succeeded without any problems. This approach can be used for any failed components.

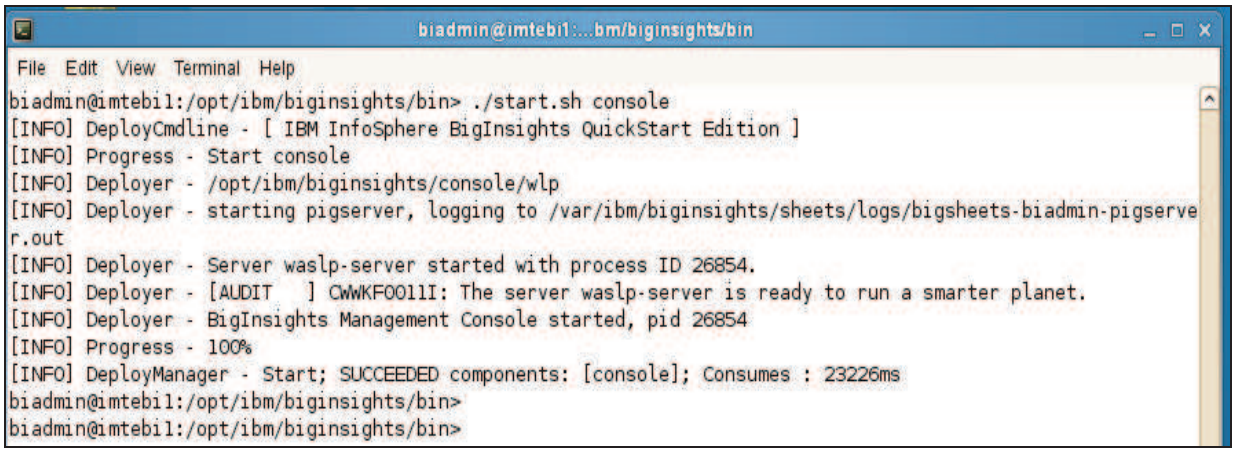


Figure 4 - Starting a specific component

Once all components have started successfully you can then move to the next section.

4 Exploring Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS), allows user data to be organized in the form of files and directories. It provides a command line interface called *FS shell* that lets a user interact with the data in HDFS accessible to Hadoop MapReduce programs.

There are two methods to interact with HDFS:

1. You can use the command-line approach and invoke the FileSystem (fs) shell using the format: `hadoop fs <args>`. This is the method we will use in this lab..
2. You can also manipulate HDFS using the BigInsights Web Console. You will explore the BigInsights Web Console on another lab.

4.1 Using the command line interface

In this part, we will explore some basic HDFS commands. All HDFS commands start with ***hadoop*** followed by ***dfs*** (distributed file system) or ***fs*** (file system) followed by a dash, and the command. Many HDFS commands are similar to UNIX commands. For details, refer to the *Hadoop Command Guide* and *Hadoop FS Shell Guide*.

We will start with the **hadoop fs –ls** command which returns the list of files and directories with permission information.

Ensure the Hadoop components are all started, and from the same Gnome terminal window as before (and logged on as *biadmin*), follow these instructions:

1. List the contents of the root directory.

```
hadoop fs -ls /
```

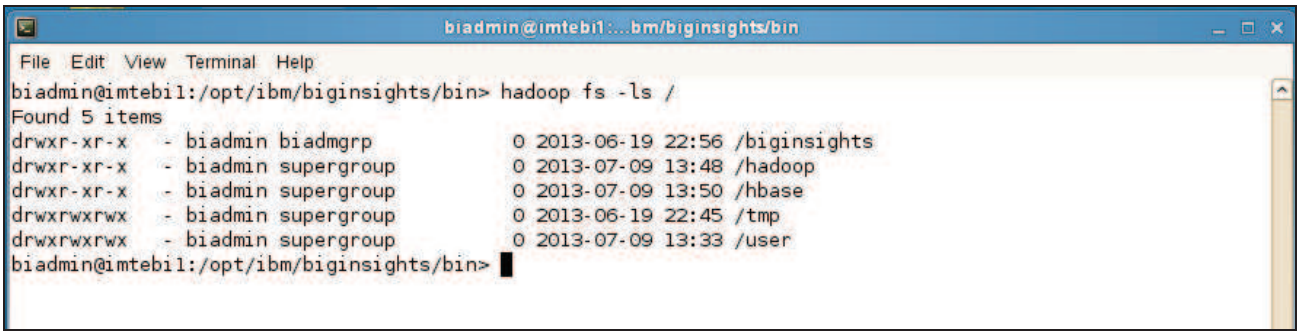


Figure 5 - List directory command

2. To list the contents of the /user/biadmin directory, execute:

```
hadoop fs -ls
```

or

```
hadoop fs -ls /user/biadmin
```

Note that in the first command there was no directory referenced, but it is equivalent to the second command where /user/biadmin is explicitly specified. Each user will get its own home directory under /user. For example, in the case of user biadmin, his home directory is /user/biadmin. Any command where there is no explicit directory specified will be relative to the user's home directory.

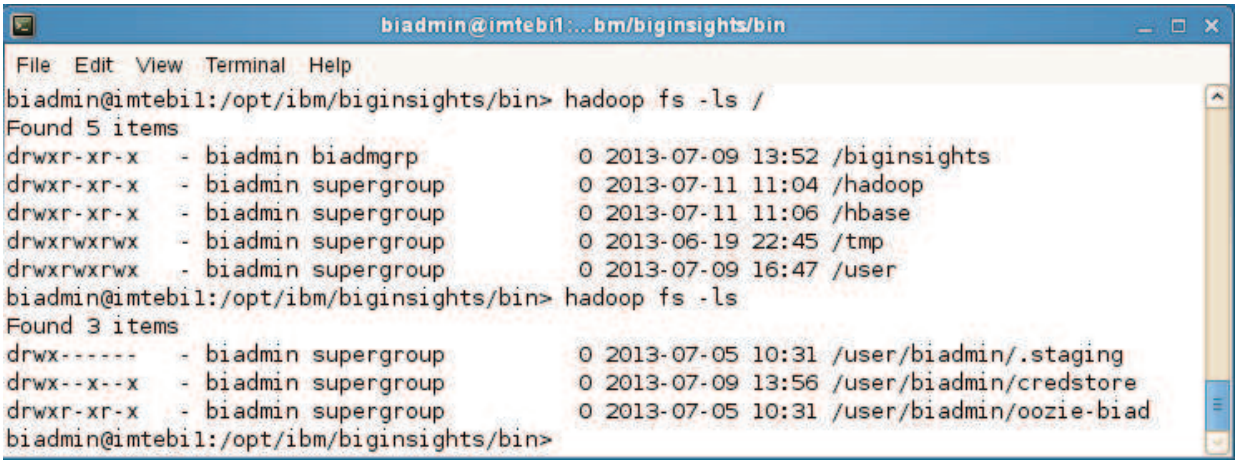


Figure 6 - hadoop fs -ls command outputs

3. To create the directory ***myTestDir*** you can issue the following command:

```
hadoop fs -mkdir myTestDir
```

Where was this directory created? As mentioned in the previous step, any relative paths will be using the user's home directory.

4. Issue the ls command again to see the subdirectory myTestDir:

```
hadoop fs -ls
```

or

```
hadoop fs -ls /user/biadmin
```

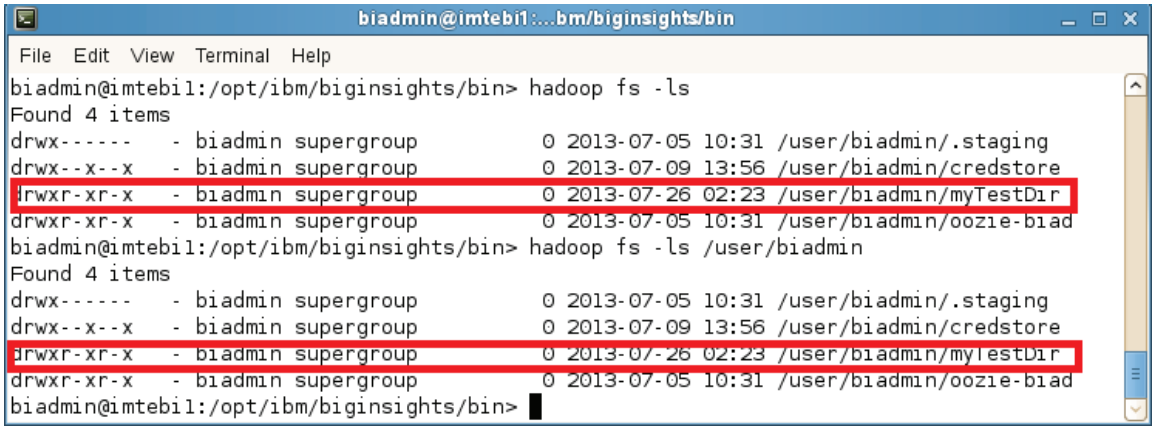


Figure 7 - hadoop fs -ls command outputs

Note: If you specify a relative path to hadoop fs commands, they will implicitly be relative to your user directory in HDFS. For example when you created the directory myTestDir, it was created in the *user/biadmin* directory.

To use HDFS commands recursively generally you add an "r" to the HDFS command (In the Linux shell this is generally done with the "-R" argument).

5. For example, to do a recursive listing we'll use the –lsr command rather than just –ls, like the examples below:

```
hadoop fs -ls /user
hadoop fs -lsr /user
```

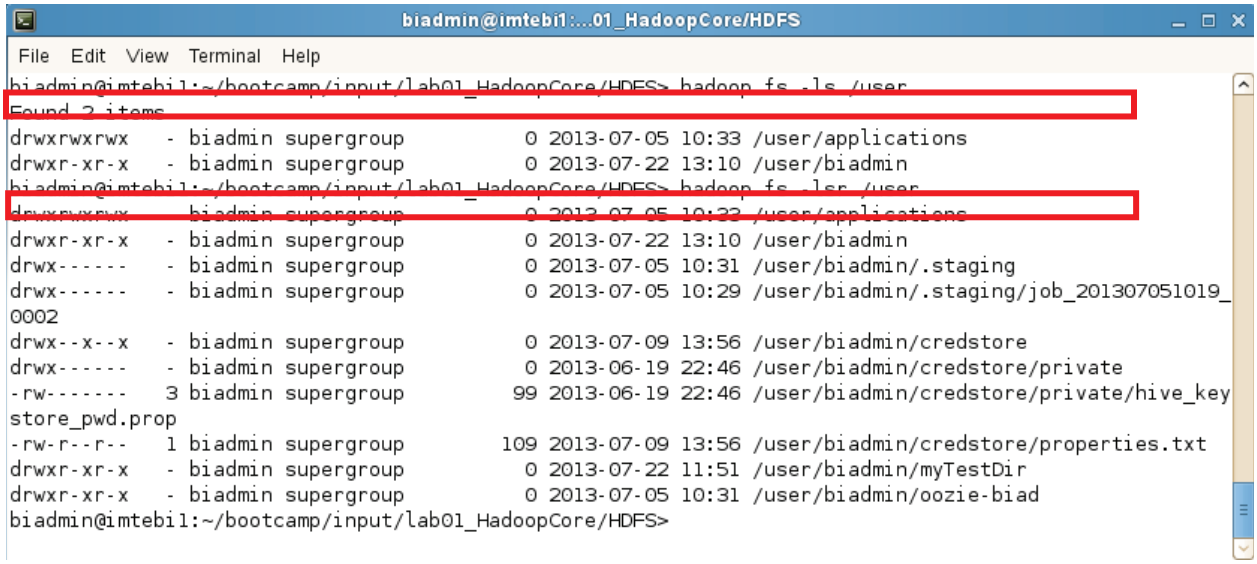


Figure 8 - hadoop fs-lsr command output

6. You can pipe (using the | character) any HDFS command to be used with the Linux shell. For example, you can easily use *grep* with HDFS by doing the following:

```
hadoop fs -mkdir /user/biadmin/myTestDir2
hadoop fs -ls /user/biadmin | grep Test
```

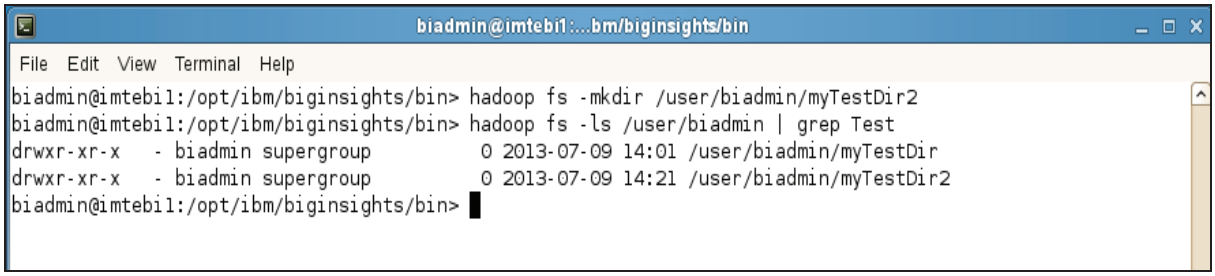


Figure 9 - hadoop fs -ls using grep to filter the output

As you can see the grep command only returned the lines which had test in them (thus removing the “Found x items” line and the .staging and oozie-biad directories from the listing

7. To move files between your regular Linux filesystem and HDFS you can use the put and get commands. For example, move the text file README to the hadoop filesystem.

```
hadoop fs -put /home/biadmin/bootcamp/input/lab01_HadoopCore/HDFS/README
README
hadoop fs -ls /user/biadmin
```

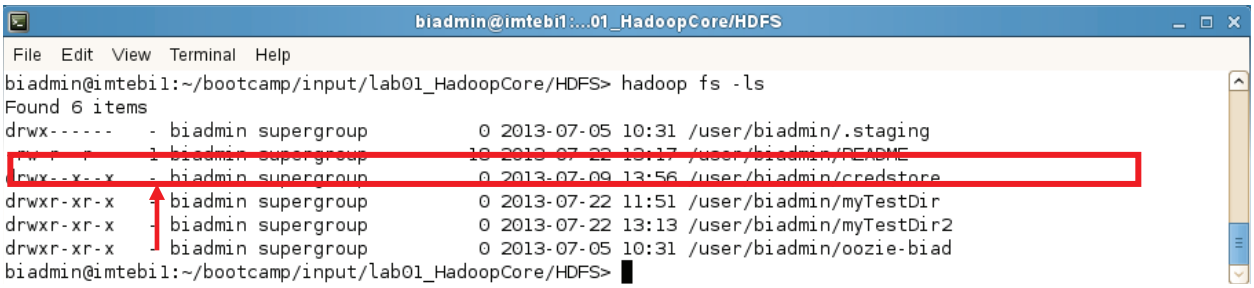


Figure 10 - README file inside HDFS

You should now see a new file called /user/biadmin/README listed as shown above. Note there is a '1' highlighted in the figure. This represents the replication factor. By default, the replication factor in a BigInsights cluster is 3, but since this laboratory environment only has one node, the replication factor is 1.

8. In order to view the contents of this file use the –cat command as follows:

```
hadoop fs -cat README
```

You should see the output of the README file (that is stored in HDFS). We can also use the linux diff command to see if the file we put on HDFS is actually the same as the original on the local filesystem.

9. Execute the commands below to use the diff command:

```
cd /home/biadmin/bootcamp/input/lab01_HadoopCore/HDFS/

diff <( hadoop fs -cat README ) README
```

Since the diff command produces no output we know that the files are the same (the diff command prints all the lines in the files that differ).

To find the size of files you need to use the –du or –dus commands. Keep in mind that these commands return the file size in bytes.

10. To find the size of the README file use the following command:

```
hadoop fs -du README
```

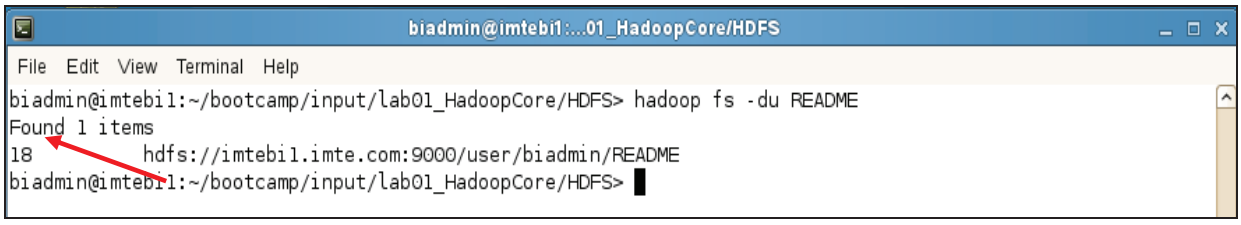


Figure 11 - Inspecting README file size

In this example, the README file has 18 bytes.

11. To find the size of all files individually in the /user/biadmin directory use the following command:

```
hadoop fs -du /user/biadmin
```

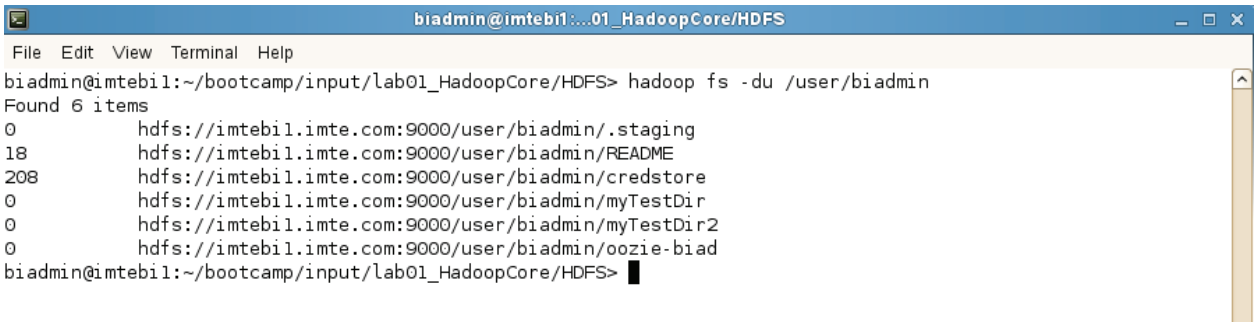


Figure 12 - Inspecting files size in a specific directory

12. To find the size of all files in total of the /user/biadmin directory use the following command:

```
hadoop fs -dus /user/biadmin
```

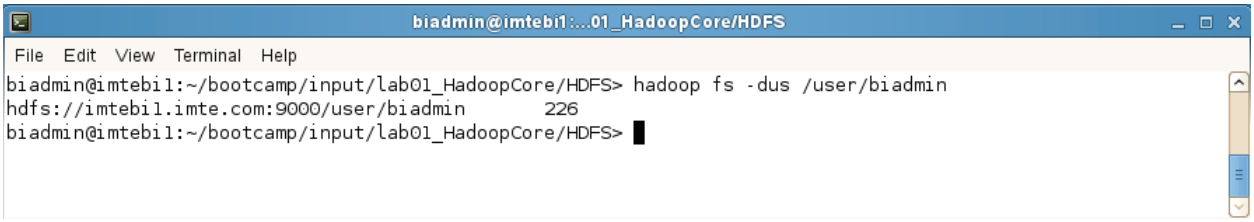


Figure 13 - Inspecting the size of directories

13. If you would like to get more information about hadoop fs commands, invoke –help as follows:

```
hadoop fs -help
```

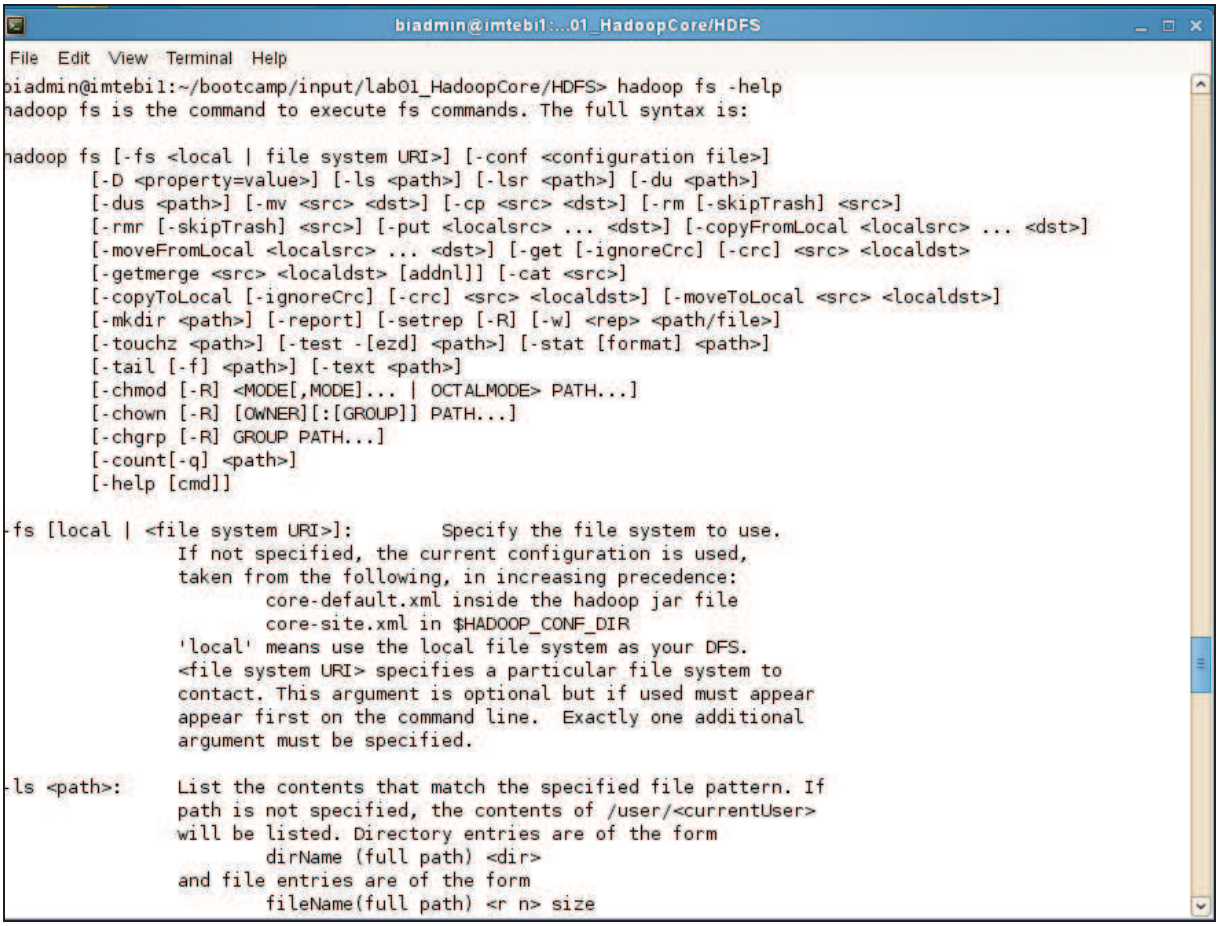


Figure 14 - Hadoop help command

14. For specific help on a command, add the command name after help. For example, to get help on the dus command you'd do the following:

```
hadoop fs -help dus
```

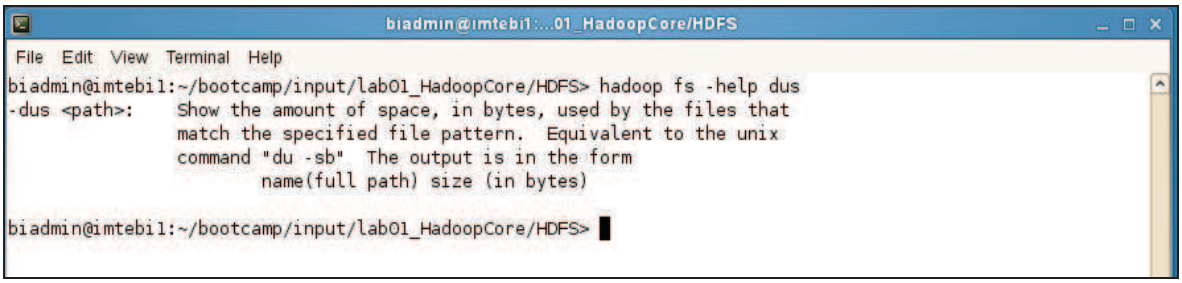


Figure 15 - Help for specific Haoop commands

5 MapReduce

Now that we've seen how the FileSystem (fs) shell can be used to execute Hadoop commands to interact with HDFS, the same *fs* shell can be used to launch MapReduce jobs. In this section, we will walk through the steps required to run a MapReduce program. The source code for a MapReduce program is contained in a compiled .jar file. Hadoop will load the JAR into HDFS and distribute it to the data nodes, where the individual tasks of the MapReduce job will be executed. Hadoop ships with some example MapReduce programs to run. One of these is a distributed WordCount program which reads text files and counts how often words occur.

5.1 Running the WordCount program

First we need to copy the data files from the local file system to HDFS.

1. Execute the commands below to copy the input files into HDFS.

```
hadoop fs -mkdir /user/biadmin/input

hadoop fs -put /home/biadmin/bootcamp/input/lab01_HadoopCore/MapReduce/*.csv
/user/biadmin/input
```

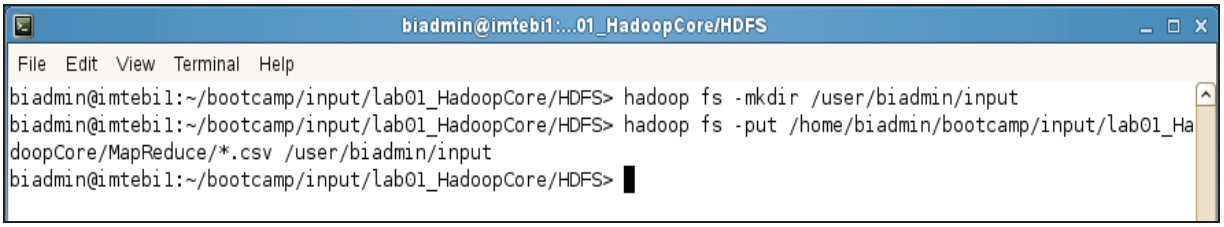


Figure 16 - Copy input files into HDFS

2. Review the files have been copied with the following command:

```
hadoop fs -ls input
```

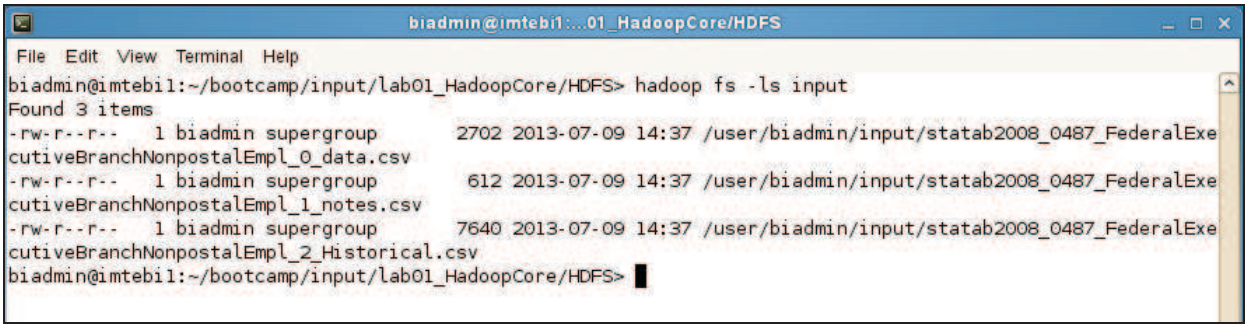


Figure 17 - List copied files into HDFS

3. Now we can run the wordcount job with the command below, where "/user/biadmin/input/" is where the input files are, and "output" is the directory where the output of the job will be stored. The "output" directory will be created automatically when executing the command below.

```
hadoop jar /opt/ibm/biginsights/IHC/hadoop-examples-1.1.1.jar wordcount
/user/biadmin/input/ output
```

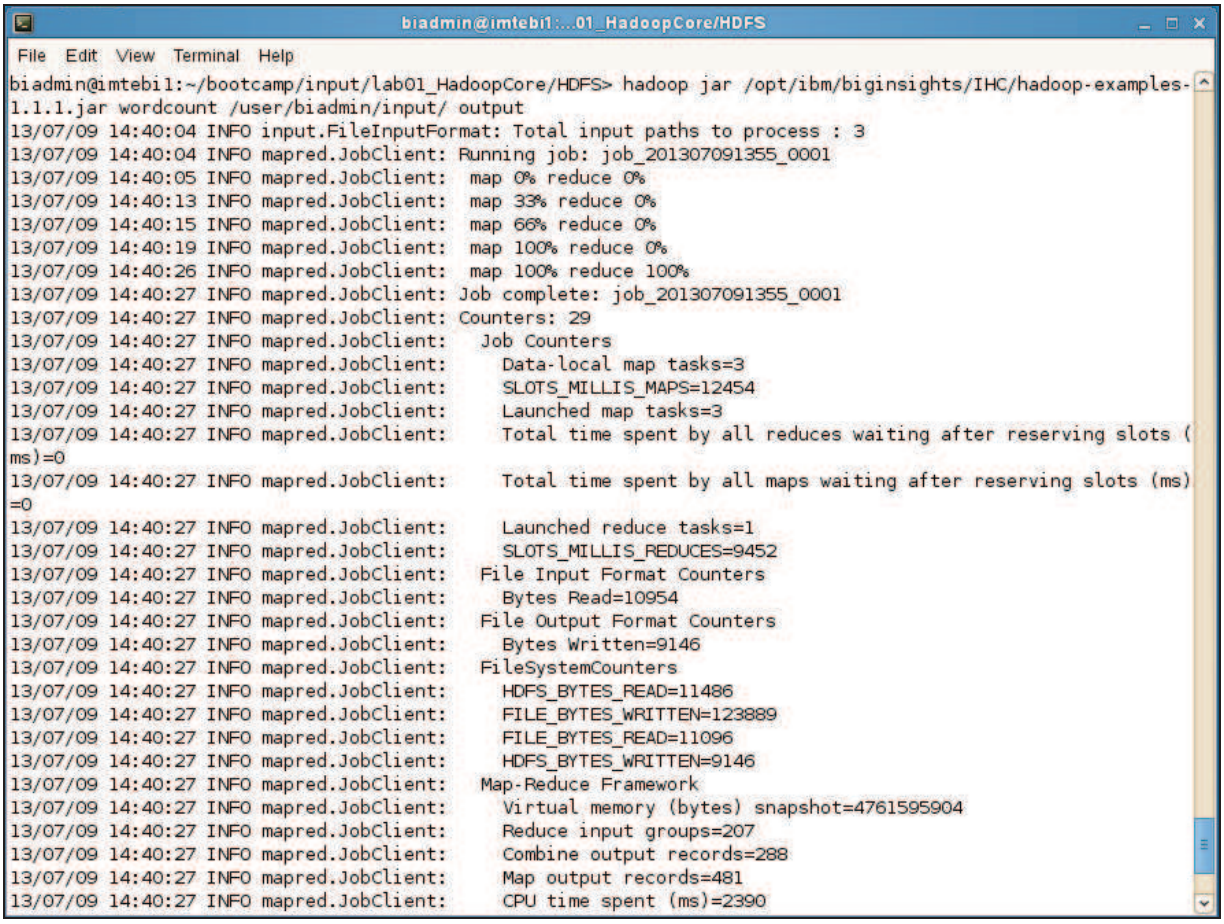


Figure 18 - WordCount MapReduce job running

4. Now review the output of step 3:

```
hadoop fs -ls output
```

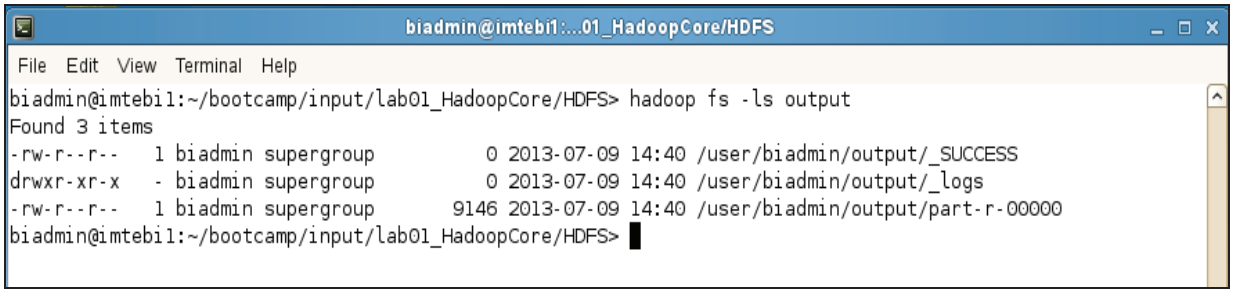


Figure 19 - MapReduce result files

In this case, the output was not split into multiple files.

5. To view the contents of the part-r-0000 file issue the command below:

```
hadoop fs -cat output/*00
```

Figure 20 - MapReduce output

Note: You can use the BigInsights Web Console to run applications such as WordCount. This same application (though with different Input files) will be run again in the lab describing the BigInsights Web Console. More detail about the job will also be described then.

6 Working with Pig

In this tutorial, we are going to use Apache Pig to process the 1988 subset of the Google Books 1-gram records to produce a histogram of the frequencies of words of each length. A subset of this database (0.5 million records) has been stored in the file **googlebooks-1988.csv** under */home/biadmin/bootcamp/input/lab01_HadoopCore/PigHiveJaql* directory.

Let us examine the format of the Google Books 1-gram records.

1. Execute the commands below to examine the format of the records:

```
cd /home/biadmin/bootcamp/input/lab01_HadoopCore/PigHiveJaql

head -5 googlebooks-1988.csv
```

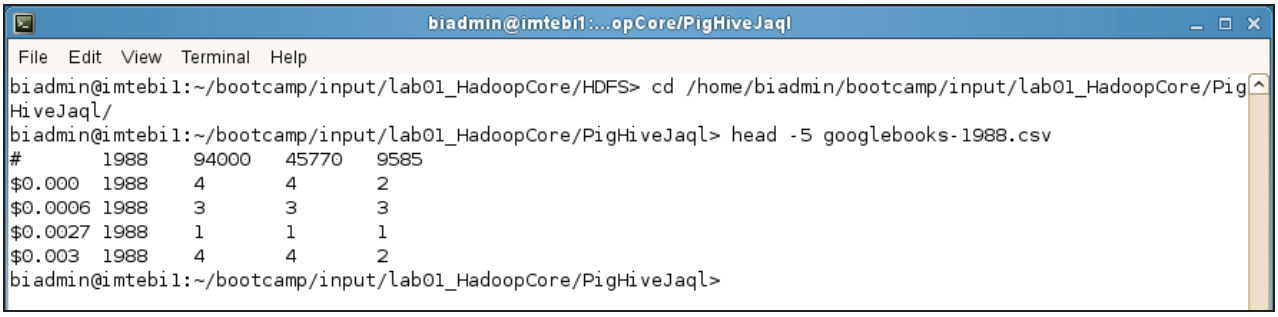


Figure 21 – Googlebooks-1988.csv file

The columns these data represent are the word, the year, the number of occurrences of that word in the corpus, the number of pages on which that word appeared, and the number of books in which that word appeared.

2. Copy the data file into HDFS.

```
hadoop fs -put googlebooks-1988.csv  pighivejaql/googlebooks-1988.csv
```

Note that directory */user/biadmin/pighivejaql* is created automatically for you when the above command is executed.

3. Start pig. If it has not been added to the PATH, you can add it, or switch to the $PIG_HOME/bin directory

```
cd $PIG_HOME/bin

./pig
```

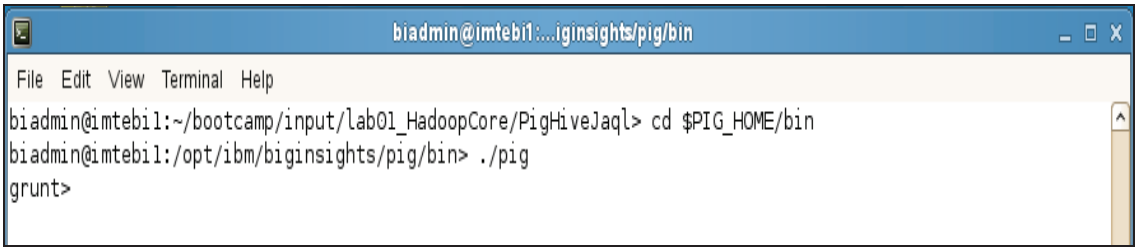


Figure 22 – Pig command line

4. We are going to use a Pig UDF to compute the absolute value of each integer. The UDF is located inside the piggybank.jar file (This jar file was created from the source, following the instructions in https://cwiki.apache.org/confluence/display/PIG/PiggyBank, and copied to the piggybank directory). We use the REGISTER command to load this jar file:

```
REGISTER  /opt/ibm/biginsights/pig/contrib/piggybank/java/piggybank.jar;
```

The first step in processing the data is to LOAD it.

5. Execute the step below to load data.

```
records = LOAD 'pighivejaql/googlebooks-1988.csv' AS (word:chararray,
year:int, wordcount:int, pagecount:int, bookcount:int);
```

   This returns instantly. The processing is delayed until the data needs to be reported.

6. To produce a histogram, we want to group by the length of the word:

```
grouped = GROUP records by
org.apache.pig.piggybank.evaluation.string.LENGTH(word);
```

7. Sum the word counts for each word length using the SUM function with the FOREACH GENERATE command.

```
final = FOREACH grouped GENERATE group, SUM(records.wordcount);
```

8. Use the DUMP command to print the result to the console. This will cause all the previous steps to be executed.

```
DUMP final;
```

   This should produce output like the following:

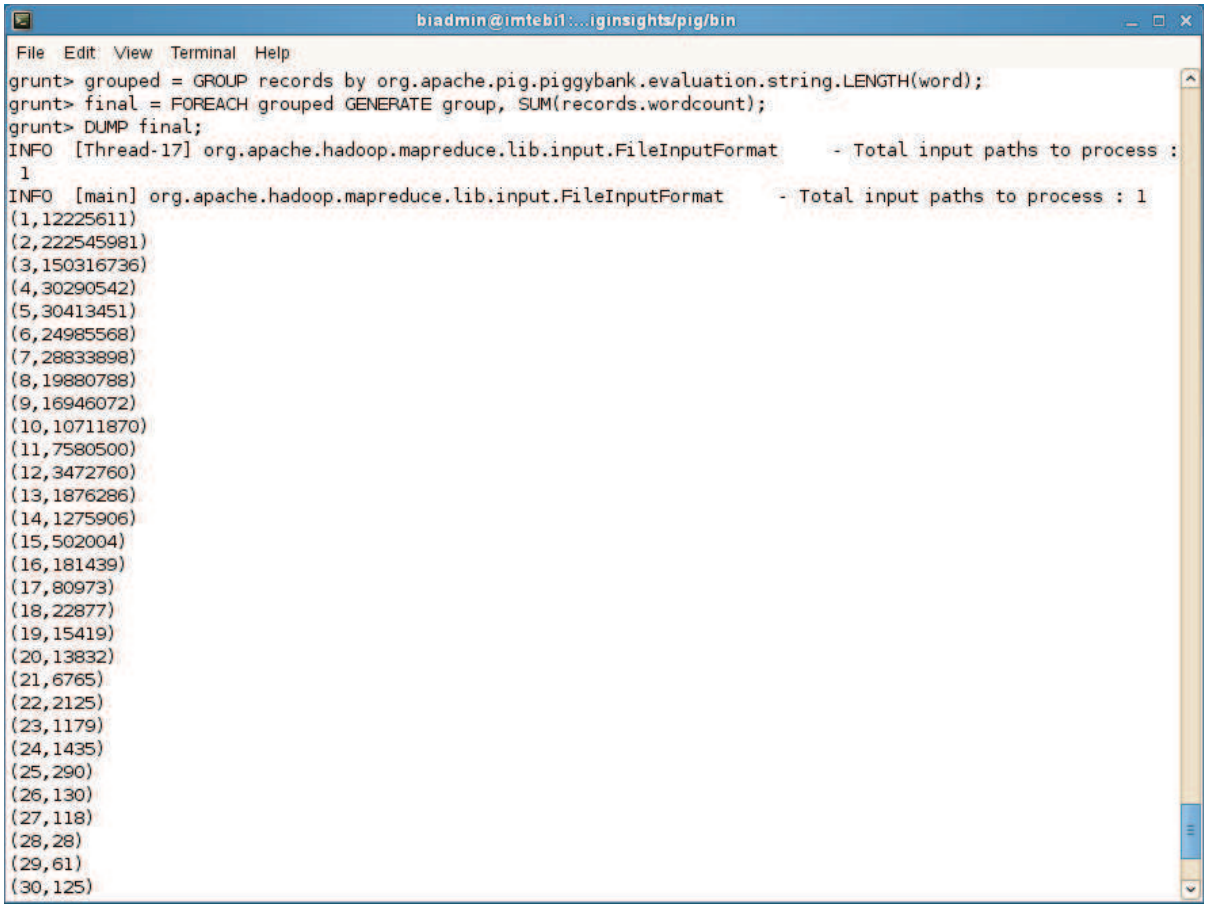


Figure 23 - Wordcount application output

9. Quit pig.

```
grunt> quit
```

7 Working with Hive

In this tutorial, we are going to use Hive to process the 1988 subset of the Google Books 1-gram records to produce a histogram of the frequencies of words of each length. A subset of this database (0.5 million records) has been stored in the file googlebooks-1988.csv under */home/biadmin/bootcamp/input/lab01_HadoopCore/PigHiveJaql* directory.

1. Ensure the Apache Derby component is started. Apache Derby is the default database used as metastore in Hive. A quick way to verify if it is started, is to try to start it using:

```
start.sh derby
```

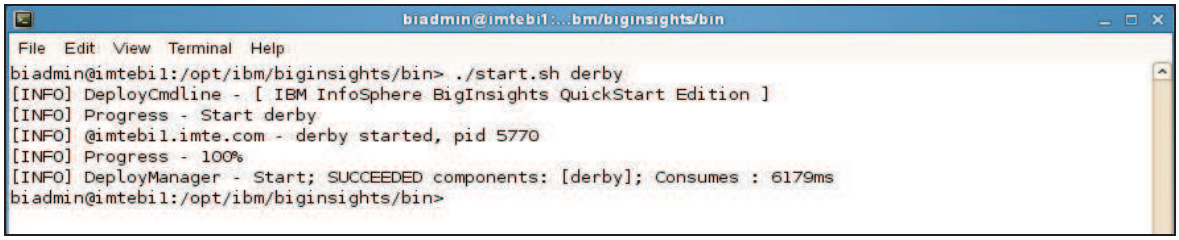


Figure 24 - Start Apache Derby

2. Start hive interactively. Change the directory to the $HIVE_HOME/bin first, and execute from there using ./hive

```
cd $HIVE_HOME/bin
./hive
```

Figure 25 - Start Apache Hive

3. Create a table called wordlist.

```
CREATE TABLE wordlist (word STRING, year INT, wordcount INT, pagecount INT,
bookcount INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

Figure 26 - Create wordlist table

4. Load the data from the googlebooks-1988.csv file into the wordlist table.

```
LOAD DATA LOCAL INPATH
'/home/biadmin/bootcamp/input/lab01_HadoopCore/PigHiveJaql/googlebooks-
1988.csv' OVERWRITE INTO TABLE wordlist;
```

Figure 27 - Load data into wordcount table

5. Create a table named wordlengths to store the counts for each word length for our histogram.

```
CREATE TABLE wordlengths (wordlength INT, wordcount INT);
```

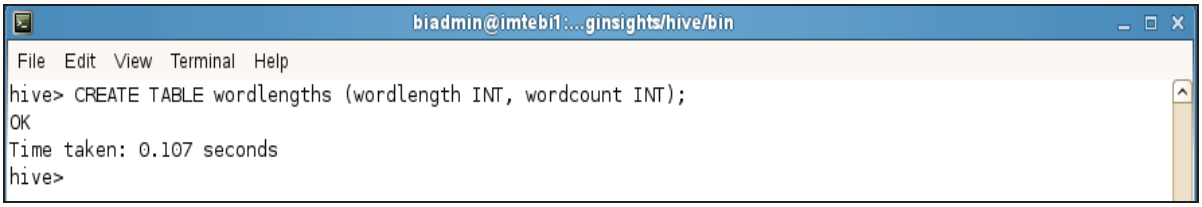


Figure 28 - Create wordlength table

6. Fill the wordlengths table with word length data from the wordlist table calculated with the length function.

```
INSERT OVERWRITE TABLE wordlengths SELECT length(word), wordcount FROM
wordlist;
```

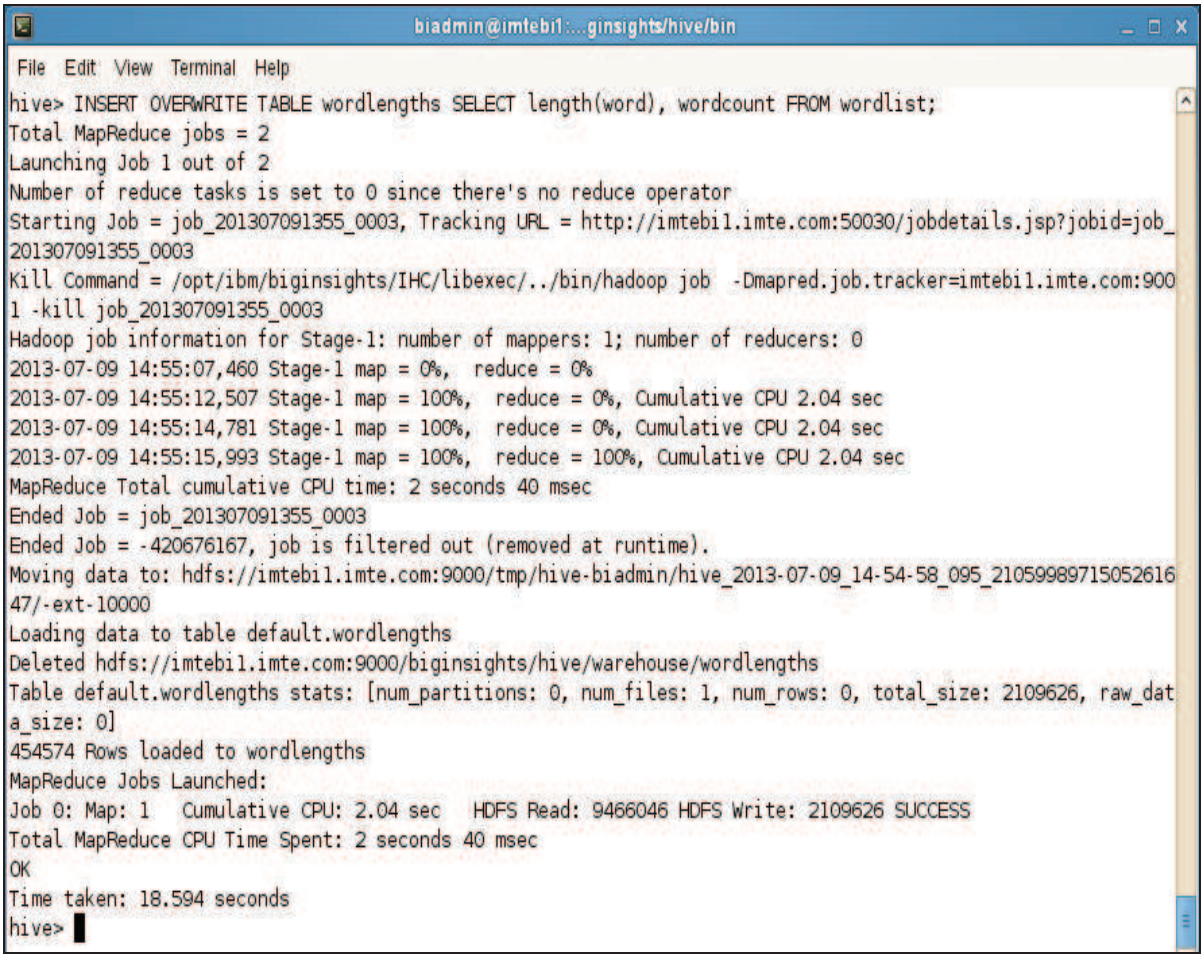


Figure 29 - Fill wordlengths table

7. Produce the histogram by summing the word counts grouped by word length.

```
SELECT wordlength, sum(wordcount) FROM wordlengths group by wordlength;
```

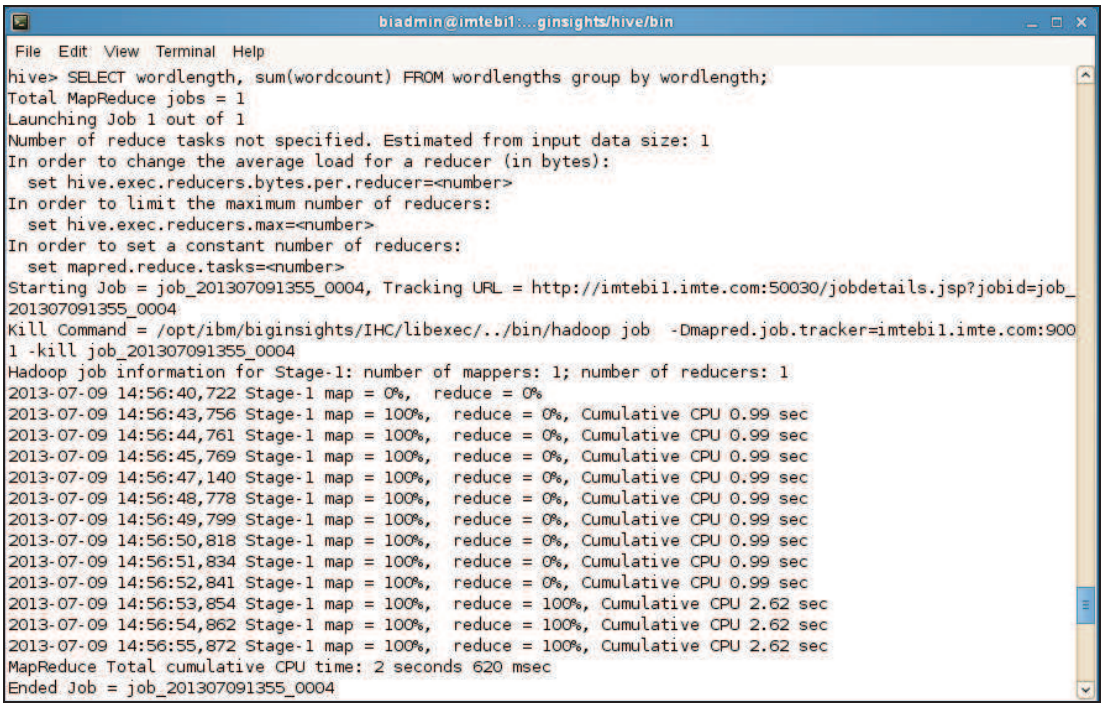


Figure 30 - Executing MapReduce job

8. Quit hive.

```
quit;
```

Figure 31 - Quit Apache Hive

8 Working with Jaql

In this tutorial, we are going to use Jaql to process the 1988 subset of the Google Books 1-gram records to produce a histogram of the frequencies of words of each length. A subset of this database (0.5 million records) has been stored in the file googlebooks-1988.csv under */home/biadmin/bootcamp/input/lab01_HadoopCore/PigHiveJaql* directory.

1. Let us examine the format of the Google Books 1-gram records:

```
cd /home/biadmin/bootcamp/input/lab01_HadoopCore/PigHiveJaql

head -5 googlebooks-1988.del
```

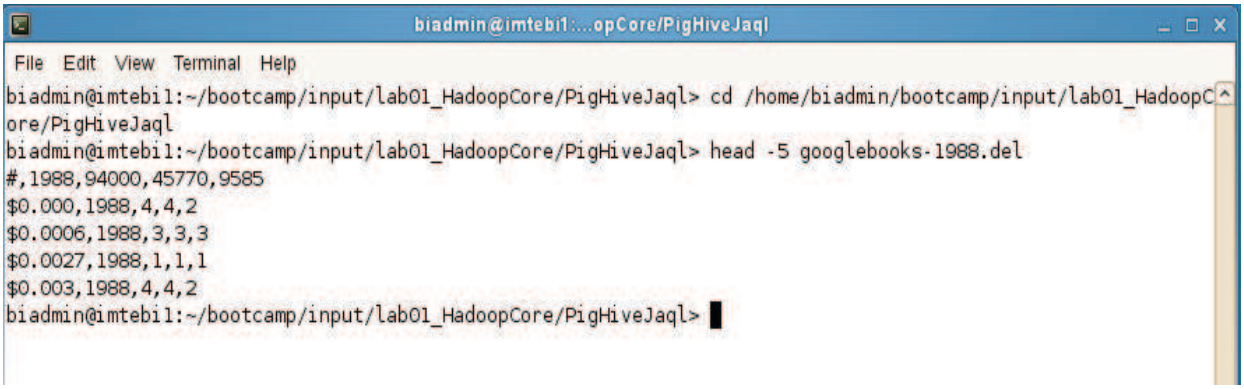


Figure 32 - googlebooks-1998.csv file format

The columns these data represent are the word, the year, the number of occurrences of that word in the corpus, the number of pages on which that word appeared, and the number of books in which that word appeared.

2. Copy the googlebooks-1988.del file to HDFS.

```
hadoop fs -put googlebooks-1988.del googlebooks-1988.del
```

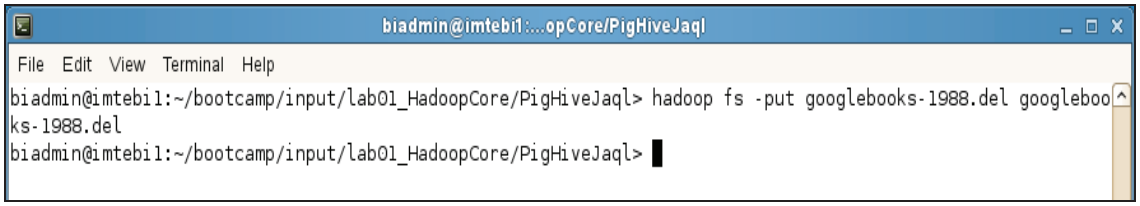


Figure 33 - Copy googlebooks-1988.csv file to HDFS

3. Change directory to $JAQL_HOME\bin, and then execute `./jaqlshell` to start the JaqlShell.

```
cd $JAQL_HOME/bin
./jaqlshell
```

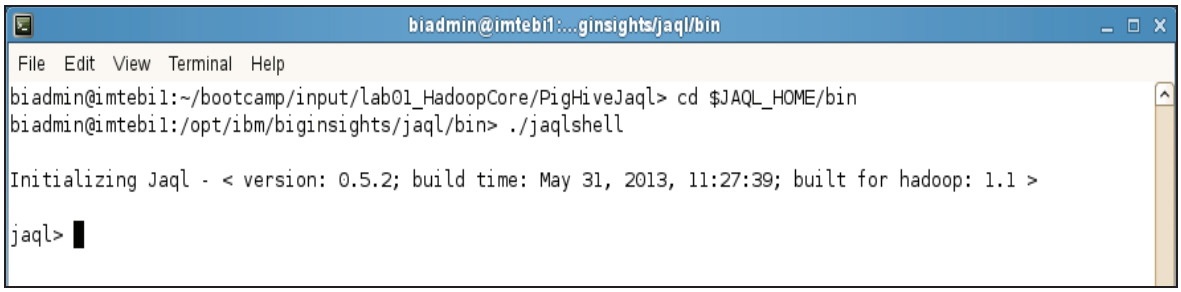


Figure 34 - Start Jaqlsell

4. Read the comma delimited file from HDFS. Note that this operation might take a few minutes to complete.

```
$wordlist = read(del("googlebooks-1988.del", { schema: schema { word:
string, year: long, wordcount: long, pagecount: long, bookcount: long } }));
```

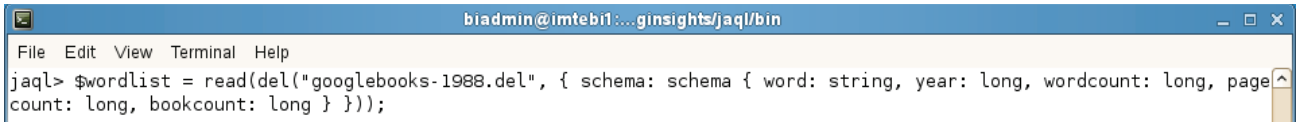


Figure 35 - Read googlebooks-1988.del from HDFS

5. Transform each word into its length by applying the strLen function.

```
$wordlengths = $wordlist -> transform { wordlength: strLen($.word),
wordcount: $.wordcount };
```

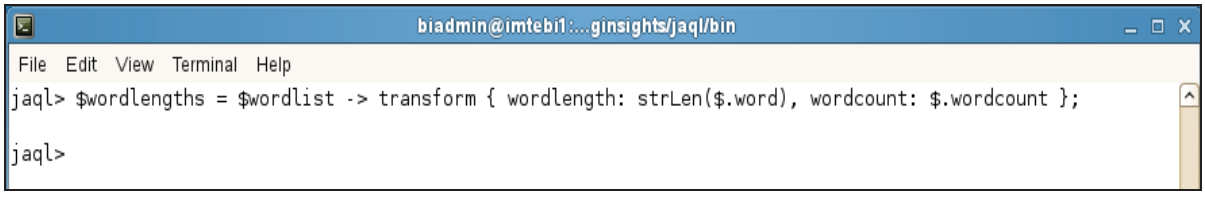


Figure 36 - Applying strLen() function

6. Produce the histogram by summing the word counts grouped by word length.

```
$wordlengths -> group by $word = {$.wordlength} into { $word.wordlength,
counts: sum($[*].wordcount) };
```

This should produce output like the following:

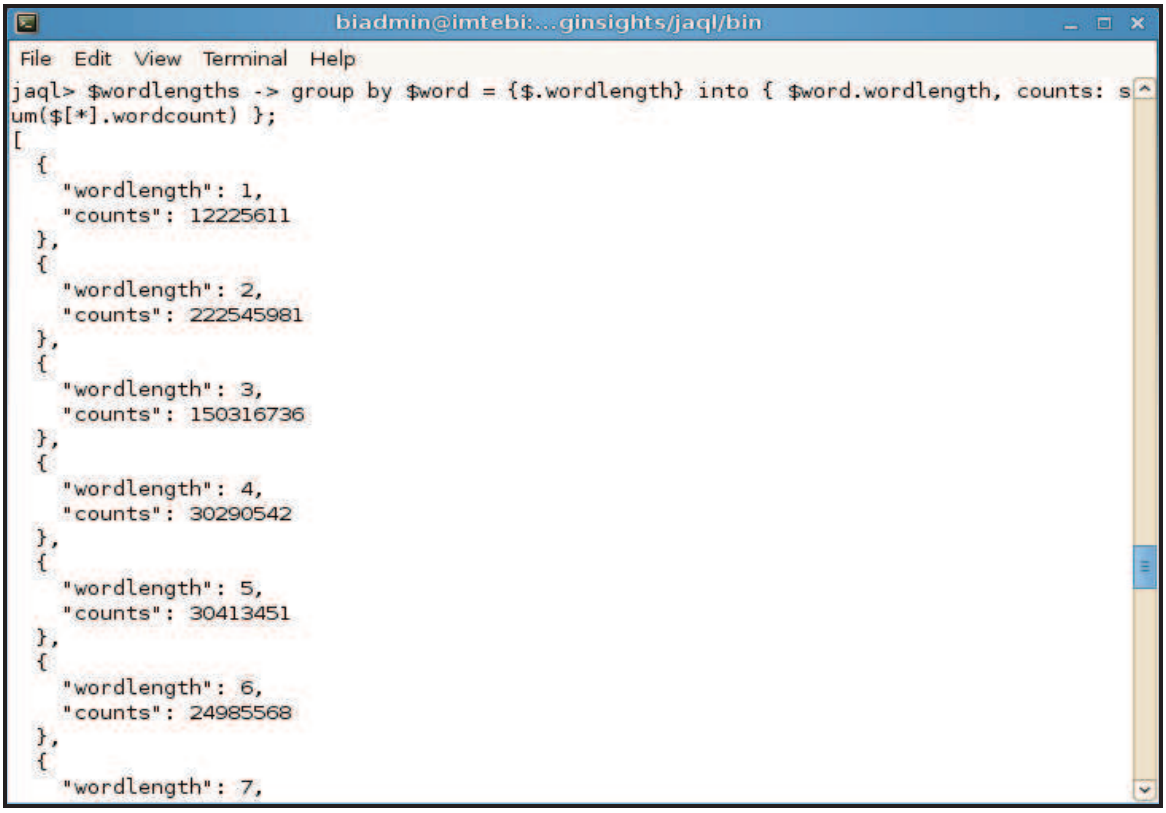


Figure 37 - Wordcount output

7. Quit Jaql.

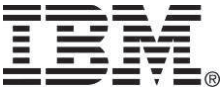
```
quit;
```

Figure 38 - Quit Jaql shell

9 Summary

You have just completed Lab 1 which focused on the basics of the Hadoop platform, including HDFS, MapReduce, Pig, Hive, and Jaql. You should now know how to perform the following basic tasks on the platform:

- Start/Stop the Hadoop components
- Interact with the data in the Hadoop Distributed File System (HDFS)
- Navigate within HDFS
- Run MapReduce programs
- Use Pig, Hive, and Jaql languages to interact with Hadoop



IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada