CMU MSP 36602 Intro to Hadoop

H. Seltman, Feb 20+25, 2019

a) Starting and Stopping Hadoop on your Ubuntu Virtual Machine

i) Hadoop is implemented as a Java process. Check which Java processes are running with `jps`. You should only see "Jps" itself.

ii) Start Hadoop and Yarn on your machine by clicking the green arrow "Start Hadoop" button on the left side of your Ubuntu console.

iii) Verify that all of the components are working by running the `jps` command again. You should see the "NameNode" and "DataNode" which are the Hadoop processes, and "NodeManager" and "ResourceManager" which are the Yarn processes. ***Now you know how to check if Hadoop is running on your machine.***

iv) To stop Hadoop, click the red "Stop Hadoop" button. The UNIX command jps can be used to verify that all of the Hadoop Java processes have been shut down. There are reports that you may run into difficulty if you do not stop Hadoop before closing VirtualBox. To close VirtualBox close the window or use the menu item File/Close. Either "saving the machine state" or "powering off" the machine are OK.

b) Getting to know the Hadoop file system

i) The Hadoop file system runs alongside the UNIX file system. We use the `hdfs` (hadoop distributed file system) command to manipulate the Hadoop file system and transfer files between systems. Hadoop analyses only run on the Hadoop file system.

ii) Think of the `hdfs` command as the "prefix" to a set of standard commands that apply to the Hadoop file system.

iii) Most of the commands of `hdfs` are technical things you will likely never need to use. The main command is `dfs`. The syntax is `hdfs dfs -subcommand [subcommand options]`. You can see valid the subcommands here. Many of the subcommands mimic standard UNIX file commands, e.g., `ls`, `mkdir`, etc., but not `cd`.

iv) ✶On your version of the Hadoop machine, your home directory on the Hadoop side is "/users/student" (as opposed to /home/student" on the UNIX side). Your working directory is your (unchangeable) home directory.

v) Here are the most useful sub commands for the `hdfs dfs` command:

(1) `ls`, e.g., `hdfs dfs -ls /user/student`

(2) `mkdir` and `rmdir`, e.g., `hdfs dfs -mkdir test`

(3) `put UNIXFile(s) HadoopFileDir`, i.e., transfer file(s) from UNIX to Hadoop, e.g.:

```
echo abc > letters.txt; echo def > letters2.txt
ls
hdfs dfs -put letters.txt letters2.txt /user/student
[or hdfs dfs -put letters.txt letters2.txt . ]
hdfs dfs -ls
```

(4) `get HadoopFile(s) UNIXDir`, i.e., transfer file(s) from Hadoop to UNIX, e.g.,

```
hdfs dfs -get /user/student/letters*.txt ~/test
```

(5) `appendToFile UNIXFile HadoopFile`, e.g., `hdfs dfs -appendToFile letters2.txt /user/student/letters.txt`

(6) `cat`, e.g., `hdfs dfs -cat letters.txt`

(7) Additional commands (with obvious syntax) include `rm, mv, cp, tail` (but not head), and `du` (disk usages).

c) Appending to ~/.bashrc to make some commands be "auto-run"

i) The UNIX file "~/.bashrc" contains important commands that are entered into your bash shells as they start up. You can add to that file (typically at the end of the file).

ii) We will be running various examples stored in a jar (Java Archive) file that comes with Hadoop. First use `which hadoop` to find where the hadoop binary lives on your system. Then replace "bin/hadoop" with "share/hadoop" and `ls` at this new location. This shows the folders for various hadoop programs. Look in the "mapreduce" folder and note the name of the .jar file containing the examples. Examine the shell variable called "mrex" that I loaded in the your auto-run file. We will need this soon.

iii) Here is another line that I put in your auto-run file: `alias dfs="hdfs dfs".` This means that when you type "dfs" it is equivalent to typing "hdfs dfs" (just to save typing). You can use the "alias" command to view all current aliases. Consider periodically adding other entries to "~/.bashrc". Note that commands you add are not run until the next time you open a terminal.

d) First Hadoop project: word counting

i) Create UNIX folder "ex1" in your home directory and `cd` to it.

ii) Make a simple, useless input dataset with `ps -aux > psaux.txt`

iii) Create folder "/user/student/ex1" on the hadoop file system using the syntax `hdfs dfs -mySubcommand myOptions` (or use the alias). Note that on the Hadoop side "ex1" points to "/user/student/ex1".

iv) Copy the psaux.txt "input" text file to "ex1" on the Hadoop file system using `hdfs dfs -put` with the appropriate source and destination.

v) Open Firefox from the Ubuntu icon, and click on the quick links bar link to "Hadoop jobs".

vi) Back in the terminal window, start the Hadoop job to use map-reduce to count the words in "psaux.txt". We will use the `hadoop` command with the first argument `jar` to tell it that we are using the option to supply the map-reduce program we want to run in the form of compiled java byte-code in a Java ARchive. As you might expect, the parameter after `hadoop jar` is the name of the jar, and you can now use your shell variable `$mrex` rather than typing that long file path. As you also might expect, what follows `hadoop jar $mrex` is any options to the examples jar. In this case the parameters are first the name of one of the many examples in the jar (`wordcount` now), followed by any specific parameters for that example (input file and output directory). So the full command you need is `hadoop jar $mrex wordcount ex1/psaux.txt ex1/out`. ***Note that the program you run is on the normal UNIX file system, while the input data and output files are on the Hadoop file system.*** (The `hadoop` command loads the code onto the Hadoop machines.)

vii) Various lines of information show on the screen (hopefully no errors). Quickly go to the Firefox window and press F5 to update the information. You may see part of what is happening if you are quick enough. You can click links to get more info. Play a bit.

viii) Return to the UNIX window, assure that you have a new prompt, and examine the ancillary data (in the terminal window) about your first map-reduce analysis.

ix) Examine your actual Hadoop results using `hdfs dfs` to see the contents of the "ex1/out" folder. You should see a file called "_SUCCESS" with size 0, which indicates success, and a file called "part-r-00000" with a size of about 10KB containing your results. Use the `hdfs dfs -get` command to get that result file into your UNIX file space. (You can use just a period for the destination to indicate "here" with the same file name.

x) Use the plain old UNIX head and tail commands to examine the copy of your results that you pulled back into the UNIX file system. Can you write a piped command to count how many words have greater than one occurrence?

xi) Use `ls -lrt psaux.txt` to see the size of "psaux.txt". Use `ps -aux >> psaux.txt` to put more text into the input file. Re-check the size. Think carefully, then carry out all steps to re-run the analysis on the new input. Note that you will need to delete the old output files or use a different directory name. Rename the old output file on the UNIX side to a different name before getting the new output file. Compare the number of words between the two files, the number of non-singlicate words, and the counts for a few specific high frequency words.

e) Second Hadoop project: word counting from your own Java program

i) Imagine that you knew how to write your own Java code. Create "/home/student/ex2" on UNIX and "/user/student/ex2 on Hadoop". Download and skim "WordCount.java". (One way to get the file is

```
wget http://www.stat.cmu.edu/~hseltman/602/code/WordCount.java
```

What other ways do you know?) Here is how to byte-compile the program to (three) class files, and then combine those into a jar file called "wc.jar":

As one-time setup, type these UNIX commands:

```
echo $JAVA_HOME
echo $PATH
```
if needed:
```
export PATH=${JAVA_HOME}/bin:${PATH}
echo $PATH

echo $HADOOP_CLASSPATH
```
if needed:
```
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
echo $HADOOP_CLASSPATH
```

Compile using:
```
hadoop com.sun.tools.javac.Main WordCount.java
ls -lrt
```

Combine the classes into a Jar using:
```
jar cf wc.jar WordCount*.class
ls -lrt
```

ii) Now think about how you could run the word count task using your jar file instead of the one from $mrex. Note that the name of the app in the wc.jar file is "WordCount". Run it.

f) Third Hadoop project: word counting in Python

i) You can look at the reference, http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/ in your browser.

ii) Note: On either the Unix or the Hadoop side "/usr/bin/python" is 2.7, "/usr/bin/python3" is 3.5, and "/usr/bin/env python" is 2.7. On the Unix side, "python" is python 3.5 via an alias.

iii) Create folder "ex3" and `cd` to it. Download mapperWC.py and reducterWC.py from 602/code. Be sure you understand the code.

iv) Try the following test code in UNIX:

```
chmod u+x mapperWC.py
echo "spam spam 602 spam eggs" | ./mapperWC.py
```

```
chmod u+x reducerWC.py
echo "spam spam 602 spam eggs" | ./mapperWC.py | sort | ./reducerWC.py
```

```
ps -aux | grep -v stringPrefs | ./mapperWC.py | sort | ./reducerWC.py
```

```
ps -aux | grep -v stringPrefs > psaux.txt
```

v) Sill on Unix, map and reduce psaux.txt. Add a pipe through "`cut`" using "`-d$'\t'`" to get just the counts. Sort the result using the `-g` flag to get a numeric sort. Pipe through "`uniq`" using the `-c` flag to count.

vi) Make "/user/student/ex3" in the Hadoop file system. Upload "psaux.txt" to Hadoop.

vii) In Unix, list the files in "/usr/local/hadoop/share/hadoop/tools/lib" restricting to those filenames containing "str". These are the Java ARchives that we can use to run our Python programs. Examine the shell variable "sjar" that I put in your auto-run.

viii) Run the Python map/reduce program on the Hadoop system as follows. This both uploads files (with -file) and sets the mapper and reducer (-mapper and -reducer).

```
hadoop jar $sjar -file mapperWC.py -mapper mapperWC.py -file
reducerWC.py -reducer reducerWC.py -input ex3/psaux.txt -output
ex3/out
```

ix) Check the results

g) Make mapper and reducer files to process 602/data/streets.txt as in IntroToBigData.pdf.