Explore RDDs with Spark Shell

Exercise Path: ~/workspace/spark-shell

Local Data Path: ~/materials/data/frostroad.txt

HDFS Data Path: weblog

**This Exercise will use the Spark Shell to work with RDDs.**

You will start by viewing and bookmarking the Spark documentation in your browser. Then you will start the Spark Shell and read a simple text file into a Resilient Distributed Data Set (RDD). Finally, you will use RDDs to transform weblog data previously imported to HDFS.

**Starting the Spark Shell**

You may choose to do the remaining steps in this exercise using either Scala or Python. Follow the instructions below for Python, or skip to the next section for Scala.

NOTE: Instructions for Python are in **blue**, and instructions for Scala are in **red**.

**Starting the Python Spark Shell**

Follow these instructions if you are using Python to complete this exercise. Otherwise, skip this section and continue with Starting the Scala Spark Shell.

4. In a terminal window, start the pyspark shell:

$ pyspark

You may get several INFO and WARNING messages, which you can disregard. If you don’t see the In[*n*]> prompt after a few seconds, hit Return a few times to clear the screen output.

1. Spark creates a SparkContext object for you called sc. Make sure the object exists:

pyspark> sc

**Note on Shell Prompt**

To help you keep track of which shell is being referenced in the

instructions, the prompt will be shown here as either pyspark> or scala>.

The actual prompt will vary depending on which version of Python or Scala

you are using and what command number you are on.

Pyspark will display information about the sc object such as:

<pyspark.context.SparkContext at 0x2724490>

1. Using command completion, you can see all the available SparkContext methods: type sc. (sc followed by a dot) and then the [TAB] key.
2. You can exit the shell by hitting control-d or by typing exit. But stay in the shell now to complete the remainder of this exercise.

**Starting the Scala Spark Shell**

Follow these instructions if you are using Scala to complete this exercise. Otherwise, skip this section and continue with Reading and Displaying a Text File.

8. In a terminal window, start the Scala Spark Shell:

$ spark-shell

You may get several INFO and WARNING messages, which you can disregard. If you don’t see the scala> prompt after a few seconds, hit Enter a few times to clear the screen output.

1. Spark creates a SparkContext object for you called sc. Make sure the object exists:

scala> sc

**Note on Shell Prompt**

To help you keep track of which shell is being referenced in the instructions, the prompt will be shown here as either pyspark> or scala>. The actual prompt will vary depending on which version of Python or Scala you are using and which command number you are on.

10. Scala will display information about the sc object such as:

res0: org.apache.spark.SparkContext =

org.apache.spark.SparkContext@2f0301fa

1. Using command completion, you can see all the available SparkContext methods: type sc. (sc followed by a dot) and then the [TAB] key.
2. You can exit the shell at any time by typing sys.exit. But stay in the shell now to complete the remainder of this exercise.

**Reading and Displaying a Text File (Python or Spark)**

1. Review the simple text file you will be using by viewing (without editing) the file in a text editor in a separate window (not the Spark shell). The file is located at:

$ ~/materials/data.

1. Define an RDD to be created by reading in the test file on the local file system. Use the first command if you are using Python, and the second one if you are using Scala.

pyspark> mydata = sc.textFile(\

"file:/home/[username]/materials/\

data/frostroad.txt")

scala> val mydata = sc.

textFile("file:/home/[username]/materials/data/f

rostroad.txt")

(Note: In subsequent instructions, both Python and Scala commands will be shown but not noted explicitly; Python shell commands are in blue and preceded with pyspark>, and Scala shell commands are in red and preceded with scala>.)

1. Spark has not yet read the file. It will not do so until you perform an operation on the RDD. Try counting the number of lines in the dataset:

pyspark> mydata.count()

scala> mydata.count()

The count operation causes the RDD to be materialized (created and populated). The number of lines (23) should be displayed, for example:

Out[4]: 23 (Python) *or*

res0: 23 (Scala)

# 

1. Try executing the collect operation to display the data in the RDD. Note that this returns and displays the entire dataset. This is convenient for very small RDDs like this one, but be careful using collect for more typical large datasets.

pyspark> mydata.collect()

scala> mydata.collect()

1. Using command completion, you can see all the available transformations and operations you can perform on an RDD. Type mydata. and then the [TAB] key.

**A Tip for PySpark Users: Controlling Log Messages**

You may have noticed that by default, PySpark displays many log messages tagged INFO. If you find this output distracting, you may temporarily override the default logging level by using the command: sc.setLogLevel("WARN"). You can return to the prior level of logging with sc.setLogLevel("INFO") or by restarting the PySpark shell. Configuring logging will be covered later in the course.

**Exploring the Lanier Web Log Files**

In this section you will be using web server log files previously imported into HDFS

using Flume.

1. Using the HDFS command line or Hue File Browser, review one of the files in the HDFS /lanier/weblogs directory, such as FlumeData.1423586038966. Note the format of the lines, for example:

IP address User ID

116.180.70.237 - 128 [15/Sep/2013:23:59:53 +0100]

GET /Html\_siteDOC-00031.html HTTP/1.0" 200 1388

Request

"http://www.lanier.com" "Lanier CSR Browser"

19. Set a variable for the data file so you do not have to retype it each time.

pyspark> logfile="/user/[username]/weblog/access\_log"

scala> var logfile="/user/[username]/weblog/access\_log"

20. Create an RDD from the data file.

pyspark> logs = sc.textFile(logfile)

scala> var logs = sc.textFile(logfile)

21. Create an RDD containing only those lines that are requests for JPG files.

pyspark> jpglogs=logs.filter(lambda line: ".jpg" in line)

scala> var jpglogs=logs.filter(line =>line.contains(".jpg"))

22. View the first 10 lines of the data using take:

pyspark> jpglogs.take(10)

scala> jpglogs.take(10)

1. Sometimes you do not need to store intermediate objects in a variable, in which case you can combine the steps into a single line of code. For instance, if all you need is to count the number of JPG requests, you can execute this in a single command:

pyspark> sc.textFile(logfile).filter(lambda line: ".jpg" in line).count()

scala> sc.textFile(logfile).filter(line => line.contains(".jpg")).count()

1. Now try using the map function to define a new RDD. Start with a simple map that returns the length of each line in the log file.

pyspark> logs.map(lambda line: len(line)).take(5)

scala> logs.map(line => line.length).take(5)

This prints out an array of five integers corresponding to the first five lines in the file.

25. That is not very useful. Instead, try mapping to an array of words for each line:

pyspark> logs.map(lambda line: line.split()).take(5)

scala> logs.map(line => line.split(' ')).take(5)

This time Spark prints out five arrays, each containing the words in the

corresponding log file line.

1. Now that you know how map works, define a new RDD containing just the IP addresses from each line in the log file. (The IP address is the first “word” in each line).

pyspark> ips = logs.map(lambda line: line.split()[0])

pyspark> ips.take(5)

scala> var ips = logs.map(line =>line.split(' ')(0))

scala> ips.take(5)

1. Although take and collect are useful ways to look at data in an RDD, their output is not very readable. Fortunately, though, they return arrays, which you can iterate through:

pyspark> for ip in ips.take(10): print ip

scala> ips.take(10).foreach(println)

28. Finally, save the list of IP addresses as a text file:

pyspark> ips.saveAsTextFile("/lanier/iplist")

scala> ips.saveAsTextFile("/lanier/iplist")

1. In the Hue file browser, list the contents of the /lanier/iplist folder. By visiting:

http://localhost:8000/filebrowser/view=/user/[username]/ lanier/iplist

**Extra Pratice**

Use RDD transformations to create a dataset consisting of the IP address and corresponding user ID for each request for an HTML file. (Disregard requests for other file types). The user ID is the third field in each log file line.

Display the data in the form *ipaddress*/*userid*, such as:

165.32.101.206/8

100.219.90.44/102

182.4.148.56/173

246.241.6.175/45395

175.223.172.207/4115

**END**