Ask Bigger Questions

CAP - Developing with Spark and Hadoop:

Homework Assignment Guide for Students

Homework: Use RDDs to Transform a Dataset

Files and Data Used in This Homework:

Exercise Directory: `$DEV1/exercises/spark-transform`

Data files: `/loudacre/weblogs/*` (HDFS)

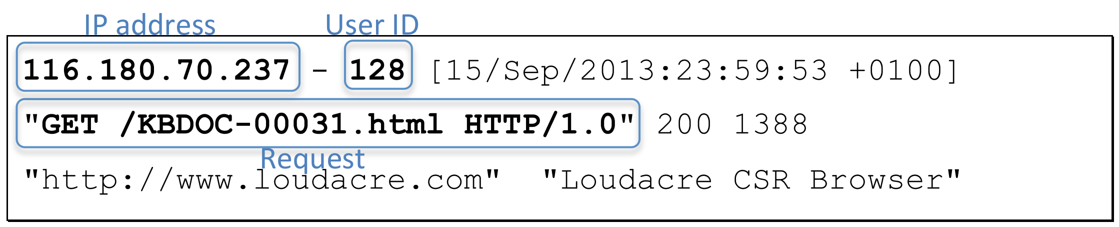
In this Exercise you will practice using RDDs in the Spark Shell.

Use Spark to explore the web server logs you captured in Flume earlier.

Explore the Loudacre web log files

In this section you will be using weblogs you imported into HDFS in the Flume exercise.

1. Using the HDFS command line or Hue File Browser, review one of the files in the HDFS `/loudacre/weblogs` directory, e.g. `FlumeData.1423586038966`. Note the format of the lines, e.g.



```
116.180.70.237 - 128 [15/Sep/2013:23:59:53 +0100]
"GET /KBDOC-00031.html HTTP/1.0" 200 1388
"http://www.loudacre.com"  "Loudacre CSR Browser"
```

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

```
pyspark> logfile="/loudacre/weblogs/FlumeData.*"
```

```
scala> var logfile="/loudacre/weblogs/FlumeData.*"
```

3. Create an RDD from the data file.

```
pyspark> logs = sc.textFile(logfile)
```

```
scala> var logs = sc.textFile(logfile)
```

4. 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"))
```

5. View the first 10 lines of the data using `take`:

```
pyspark> jpglogs.take(10)
```

```
scala> jpglogs.take(10)
```

6. 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()
```

7. 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.

8. 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 it prints out five arrays, each containing the words in the corresponding log file line.

9. 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)
```

10. 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)
```

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

```
pyspark> ips.saveAsTextFile("/loudacre/iplist")
```

```
scala> ips.saveAsTextFile("/loudacre/iplist")
```

12. In a terminal window or the Hue file browser, list the contents of the `/loudacre/iplist` folder. You should see multiple files, including several `part-`*`xxxxx`* files, which are the files containing the output data. ("Part" (partition) files are numbered because there may be results from multiple tasks running on the cluster; you will learn more about this later.) Review the contents of one of the files to confirm that they were created correctly.

Optional Homework

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`*, e.g.:

```
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
…
```

This is the end of the Homework