WORKING WITH RDD [PART A]

RDDS are immutable Resilient Distributed Dataset which are used for semi or unstructured datasets, to process and cleanse the data and we can convert the same to DataFrame. RDDs do not have schema and do not use optimizer. We create RDD using "sc" or "spark.sparkcontext" objects.

ACTION COMMANDS

COUNT
FIRST
TAKE
COLLECT
saveAsTextFile

TRANSFORMATION FUNCTIONS

MAP
FILTER
FLATMAP
DISTINCT
UNION

We use lambda anonymous function for transformation functions.

Read and Display Data from a Text File

STEP 1

Review the simple text file you will be using by viewing (without editing) the file in a separate window (not the Spark shell). The file is **frostroad.txt**. Put it on the Desktop of the VM machine.

In a terminal window from Desktop, upload the text file to HDFS directory /user/root
[root@saispark ~]# **cd Desktop/**
[root@saispark Desktop]# **hdfs dfs -put frostroad.txt /user/root/**

STEP 2

In the Spark shell, define an RDD based on the frostroad.txt text file.

On Scala

scala> **val myRDD = sc.textFile("/devsh_loudacre/frostroad.txt")**

or
scala> **val myRDD = spark.sparkContext.textFile("/user/root/frostroad.txt")**

On Python
pyspark> **myRDD = sc.textFile("/user/root/frostroad.txt")**

or
pyspark> myRDD = **Spark.sparkContext.textFile("/user/root/frostroad.txt")**

STEP 3
Using command completion, you can see all the available transformations and actions you can perform on an RDD. Type myRDD. and then the TAB key.
Spark has not yet read the file. It will not do so until you perform an action on the RDD. Try counting the number of elements in the RDD using the count action

On Scala
scala> **myRDD.count**

On Python
pyspark> **myRDD.count()**
The count operation causes the RDD to be materialized (created and populated).

STEP 4
Call the collect operation to return all data in the RDD to the Spark driver. Take note of the type of the return value; in Python will be a list of strings, and in Scala it will be an array of strings.

Note: collect returns the entire set of data. This is convenient for very small RDDs like this one, but be careful using collect for more typical large sets of data.

On Scala
scala> **val lines = myRDD.collect**

On Python
pyspark> **lines = myRDD.collect()**

STEP 5

Display the contents of the collected data by looping through the collection.

On Scala

scala> **for (line <- lines) println(line)**

On Python

pyspark> **for line in lines: print line** (press enter)
..... (Press enter again)

Transform Data in an RDD

In this exercise, you will load two text files containing the names of various cell phone makes, and append one to the other. Review the two text files you will be using by viewing (without editing) the file in a separate window. The files are makes1.txt and makes2.txt copied on the Desktop of the VM.

STEP 1

Upload the two text file to HDFS directory **/user/root/**

[root@saispark ~]# **cd Desktop/**
[root@saispark Desktop]# **hdfs dfs -put makes*.txt /user/root/**

STEP 2

In Spark, create an RDD called makes1RDD based on the makes1.txt file.

On Scala

scala> **val makes1RDD = sc.textFile("/user/root/makes1.txt")**

Display the contents of the makes1RDD data using collect and then looping through returned collection

scala> **for (make <- makes1RDD.collect()) println(make)**

Repeat the previous steps to create and display an RDD called makes2RDD based on the second file, /user/root/makes2.txt.

Create a new RDD by appending the second RDD to the first using the union transformation.

scala> **val allMakesRDD = makes1RDD.union(makes2RDD)**

Collect and display the contents of the new allMakesRDD RDD

Use the distinct transformation to remove duplicates from allMakesRDD. Collect and display the contents to confirm that duplicate elements were removed.

scala> **val distinctMakesRDD = allMakesRDD.distinct**
scala> **for (make <- distinctMakesRDD.collect()) println(make)**

On Python

In Spark, create an RDD called makes1RDD based on the makes1.txt file.

pyspark> **makes1RDD = sc.textFile("/user/root/makes1.txt")**

Display the contents of the makes1RDD data using collect and then looping through returned collection

pyspark> **for make in makes1RDD.collect(): print make**

Repeat the previous steps to create and display an RDD called makes2RDD based on the second file, /user/root/makes2.txt.

Create a new RDD by appending the second RDD to the first using the union transformation.

pyspark> **allMakesRDD = makes1RDD.union(makes2RDD)**

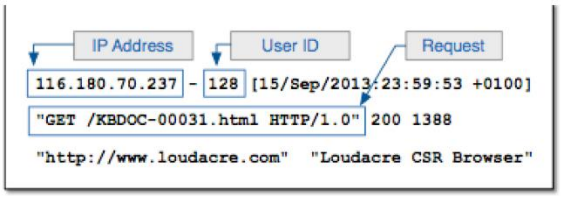
Collect and display the contents of the new allMakesRDD RDD

Use the distinct transformation to remove duplicates from allMakesRDD. Collect and display the contents to confirm that duplicate elements were removed.

pyspark> **distinctMakesRDD = allMakesRDD.distinct()**
pyspark> **for make in distinctMakesRDD.collect(): print make**

Transforming Data Using RDDs

In this section you will be using data weblogs directory. Review one of the .log files in the directory. Note the format of the lines



STEP 1

Copy the weblogs directory from the local filesystem to the /user/root/ HDFS directory.

```
[root@saispark ~]# cd  Desktop/
[root@saispark Desktop]# hdfs  dfs  -put  weblogs  /user/root/
[root@saispark Desktop]# hdfs  dfs  -ls   /user/root/weblogs/
```

On Scala

```
// Display the first 10 lines which are requests for JPG files

scala> val jpglogsRDD=logsRDD.filter(line => line.contains(".jpg"))

scala> val jpgLines = jpglogsRDD.take(5)

scala> jpgLines.foreach(println)


// Create an RDD of the length of each line in the file and display
   the first 5 line lengths


scala> val lineLengthsRDD = logsRDD.map(line => line.length)

scala> lineLengthsRDD.take(5).foreach(println)


// Map the log data to an RDD of arrays of the fields on each line

scala> val lineFieldsRDD = logsRDD.map(line => line.split(' '))

scala> val lineFields = lineFieldsRDD.take(5)

scala> for (fields <- lineFields) {

  println("-------")

  fields.foreach(println)

}
```

// **Map the log data to an RDD of IP addresses for each line**

scala> **val ipsRDD = logsRDD.map(line => line.split(' ')(0))**

scala> **ipsRDD.take(5).foreach(println)**

// **Loop through the array returned by take**

scala> **ipsRDD.take(10).foreach(println)**

// **Save the IP addresses to text file(s)**

scala> **ipsRDD.saveAsTextFile("/devsh_loudacre/iplist/")**

// **Save "ip-address,user-id"**

scala> **var userIPRDD=logsRDD.filter(_.contains(".html")).map(line =>**
line.split(' ')(0) + "," + line.split(' ')(2))
scala> **userIPRDD.saveAsTextFile("/devsh_loudacre/userips_csv")**

// **Read into a DataFrame**

scala> **val useripsDF = spark.read.option("inferSchema","true").csv("/devsh_loudacre/userips_csv")**
scala> **useripsDF.printSchema**

scala> **useripsDF.show**

On Python

Display the first 5 lines that are requests for JPG files

pyspark> **jpglogsRDD = logsRDD.filter(lambda line: ".jpg" in line)**

pyspark> **jpgLines = jpglogsRDD.take(5)**

pyspark> **for line in jpgLines: print line**

 # Create an RDD of the length of each line in the file and display the first 5 line lengths

pyspark> **lineLengthsRDD = logsRDD.map(lambda line: len(line))**

pyspark> **for length in lineLengthsRDD.take(5): print length**

 # Map the log data to an RDD of arrays of the fields on each line

pyspark> **lineFieldsRDD = logsRDD.map(lambda line: line.split(' '))**

pyspark> **lineFields = lineFieldsRDD.take(5)**

pyspark> **for fields in lineFields: print "-------"**

pyspark> **for field in fields: print field**

 # Map the log data to an RDD of IP addresses for each line, Display results

pyspark> **ipsRDD = logsRDD.map(lambda line: line.split(' ')[0])**

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

 # Save the IP addresses to text file(s)

pyspark> **ipsRDD.saveAsTextFile("/devsh_loudacre/iplist/")**

 # Map web logs to "ip-address,user-id" and save

pyspark> **userIPRDD=logsRDD.filter(lambda line: ".html" in line).map(lambda line: line.split(' ')[0]+","+line.split(' ')[2])**

pyspark> **userIPRDD.saveAsTextFile("/devsh_loudacre/userips_csv")**

 # Read the CSV data into a DataFrame

pyspark> **useripsDF = spark.read.option("inferSchema","true").csv("/devsh_loudacre/userips_csv")**

pyspark> **useripsDF.printSchema()**

pyspark> **useripsDF.show()**