Lab 2

Introduction

In this lab, you will use the Spark shell to explore variables and strings, and transform a small dataset using RDDs and core Spark with Scala.

Objectives

- Practice more Scala commands.
- Practice transferring large files to the Peel cluster.
- Practice basic RDD transformations and actions.
- Practice loading datasets and storing results in HDFS.

Documentation

First, familiarize yourself with online Spark documentation by checking out https://spark.apache.org/docs/2.4.0/.

- From the **Programming Guides** menu at the top of the page, select **RDDs, Accumulators, Broadcasts Vars** and scan through it. You can skip sections that we haven't covered yet.
- From the **API Docs** menu at the top of the page, select **Scala** and scan the material.

Your tasks

Part 1: Explore Scala variables

1. On the Peel cluster, start the Spark shell.
   ```
   spark-shell --deploy-mode client --driver-java-options=-Dscala.color
   ```
2. Create an **immutable** variable named `exchangeRate` with **explicit** type `Double` and assign to it the value `0.87`.

3. Create an **immutable** variable named `dollars` with **explicit** type `Int` and assign to it the value `100.00`. What error do you see?
4. Correct Step 3 to get rid of the error.
5. Create a **mutable** variable named `euros` with **implicit** type `Double` initialized to zero.
6. Assign to `euros` the result of converting `dollars` to euros using `exchangeRate` as the conversion factor. (*Hint: 1 US dollar equals 0.87 euro.*)
7. Assign to `dollars` a new value: 500. What error do you see?
8. Fix the error in Step 7 and set `dollars` to 500.
9. Create a new **mutable** variable, `eurosInt`, of type `Int` and assign to it 0.
10. Assign to `eurosInt` the result of converting `dollars` to euros using `exchangeRate`. What error do you see?
11. Use `toInt` to remove the error in Step 10.
12. Use `getClass` to verify the types of the three variables.
13. Output the result using the `println` command. Note that `$$` in an f-String yields a dollar sign.
    ```
    println(f"February 2022: $$$dollars = $eurosInt euros")
    ```
14. Take screenshot(s) of everything you have done, including all commands and outputs, and put them in a folder named `part1`.

Part 2: Explore Scala strings

1. Create an **immutable** variable called `record` and assign to it the following string.
   ```
   2022-02-10:19:10:00, 12345678-aaaa-1000-gggg-000111222333, 58, TRUE, enabled
   ```
2. Use `record.length` to determine the number of characters in `record`.
3. Use the `contains` method to search for the word `disabled` in `record`.
4. Use `indexOf` to find the index of the first occurrence of `,` in `record`.
5. Convert `record` to lower case using `toLowerCase` and then use chaining with `indexOf` to find the start of substring `true`.
6. Verify that Step 5 **did not** modify the variable named `record`.

7. Create a new **mutable** variable called `record2` and assign it to the contents of `record`.
8. Test whether `record == record2`. Note that `==` in Scala behaves like `equals` in Java.
9. Set `record2 = "something else"`.
10. Test whether `record == record2`.
11. Quit the Spark shell.
12. Take screenshot(s) of everything you have done, including all commands and outputs, and put them in a folder named `part2`.

Part 3: Transform a small dataset using RDDs and core Spark with Scala

1. Download 2014-03-15.log and transfer it to the Peel server. Please refer to this documentation on how to transfer data to/from HPC systems.

   Each line of the file has the following format:

   > *IP Address:* ***116.180.70.237***
   > *Separator: -*
   > *User ID: 128*
   > *Date/Time: [15/Sep/2013:23:59:53 +0100]*
   > *Request: "GET /titanic_2400.**jpg** HTTP/1.0" 200 6021*
   > *"http://www.loudacre.com" "Loudacre Mobile Browser iFruit 5"*

2. Create a directory in HDFS called `loudacre/weblog` and put the log file into that directory.
   1. Show the command you use to create the HDFS directory.
   2. Show the command you use to put the file into HDFS.

3. Start the Spark shell and show the commands you use for all of the following steps.

    1. Store the full file path `/user/yourNetID/loudacre/weblog/2014-03-15.log` to an immutable String variable named `logfile`.
    2. Create an RDD named `originalRdd` from the file.
    3. Print 5 lines of the data.
    4. Create an RDD named `jpgRdd` containing only lines that are requests for `jpg` files.
    5. Print 5 lines of the data.
    6. Chain the previous commands into a single command that counts the number of JPG requests.
    7. Create an RDD named `ipRdd` containing only the **IP address** from each line of the **original** log file.
    8. Save the list of IP addresses to an HDFS directory named `loudacre/iplist` using `saveAsTextFile`.
    9. Quit the Spark shell.

4. List the `loudacre/iplist` directory in HDFS and take a screenshot. Note that you may see multiple files, including several `part-xxxxx` files, which are the files containing the output data. “Part” files are numbered because there may be results from multiple tasks running in the cluster (the tasks are part of your Spark job).
5. Review the contents of one of the files to confirm that they were created correctly.
6. Take screenshot(s) of everything you have done, including all commands and outputs, and put them in a folder named `part3`. For the last step (Step 5), you only need to provide a screenshot of some lines of a `part-xxxxx` file that was written into the `loudacre/iplist` directory.

Submission

To receive full credit, please compress all screenshots into a single file named `YourName_NetID_lab2.zip` and upload it to NYU Brightspace.

- Your submission should contain three folders: `part1`, `part2`, and `part3`, each containing the screenshots for the respective part.

Tips

Please use Discord if you experience any difficulties. The graders and I will help you get your environment working.

Don’t procrastinate. The Hadoop cluster tends to get crowded near the due date.