# CSP554—Big Data Technologies

## Assignment #7

## Worth: 12 points (2 points for each problem)

Readings:

The mid-term will assume you have read at least the high points of the below.

In “Free Books and Chapters” section:

* Make sure to read/skim “Spark - The Definitive Guide (Excerpts)”
* Make sure to read/skim “Spark - Python API - SQL & DataFrames”
* Read “Learning Spark,” Ch. 4
* Also take a look at the two Spark Cheat Sheets

In the “Articles” section:

* Read “The Lambda and the Kappa”

For this assignment you will be using your EMR environment including Spark and the pyspark CLI.

Some basic notes:

* We will again be using files generated by the program TestDataGen. But even though the files this program generates end is the ‘.txt’ suffix, I want you to treat them as if they were (comma separated) ‘.csv’ files.
* In fact, if you like, when you copy them to HDFS you can change their suffixes from ‘.txt’ to ‘.csv’. But this is not necessary to complete the exercises.

Demos

To sets of demo files have been included in this assignment. It is a good idea to look at and execute them before trying the following exercises.

* One set of demo files provide examples of the use of RDDs. The demo files are in pydemo.zip. Use scp to copy pyspark.zip to your EMR master node (to /home/hadoop) and then follow the instructions below:

Instructions for this demo...

0) Use scp to copy pydemo.zip to your EMR master node

1) Expand the zip file using:

unzip pydemo.zip

2) cd to the Linux directory /home/hadoop/pydemo

3) First copy (using "hadoop fs ...") the files cs595doc2.txt and twinkle.txt to HDFS into the directory /user/hadoop

4) Now start pyspark

5) Execute test.py from within pyspark using:

exec(open("/home/hadoop/pydemo/test.py").read())

6) Do the same for test2.py, test3.py, test4.py, testt.py and twinkle1.py

* Another set of demo files provide examples of the use of DataFrames. the demo files are in sparkdf.zip. Use scp to copy sparkdf.zip to your EMR master node (to /home/hadoop) and then follow the instructions below:

Demo Instructions

0) Use scp to copy sparkdf.zip to your EMR master node

1) Expand the zip file using:

unzip sparkedf.zip

2) cd to the Linux directory /home/hadoop/sparkdf

3) Copy the files (using "hadoop fs ...") people.csv, peopleh.csv, people.txt and people.json into the HDFS directory /user/hadoop

4) Start pyspark

5) Execute spark1.py from within pyspark using:

exec(open("/home/hadoop/sparkdf/spark1.py").read())

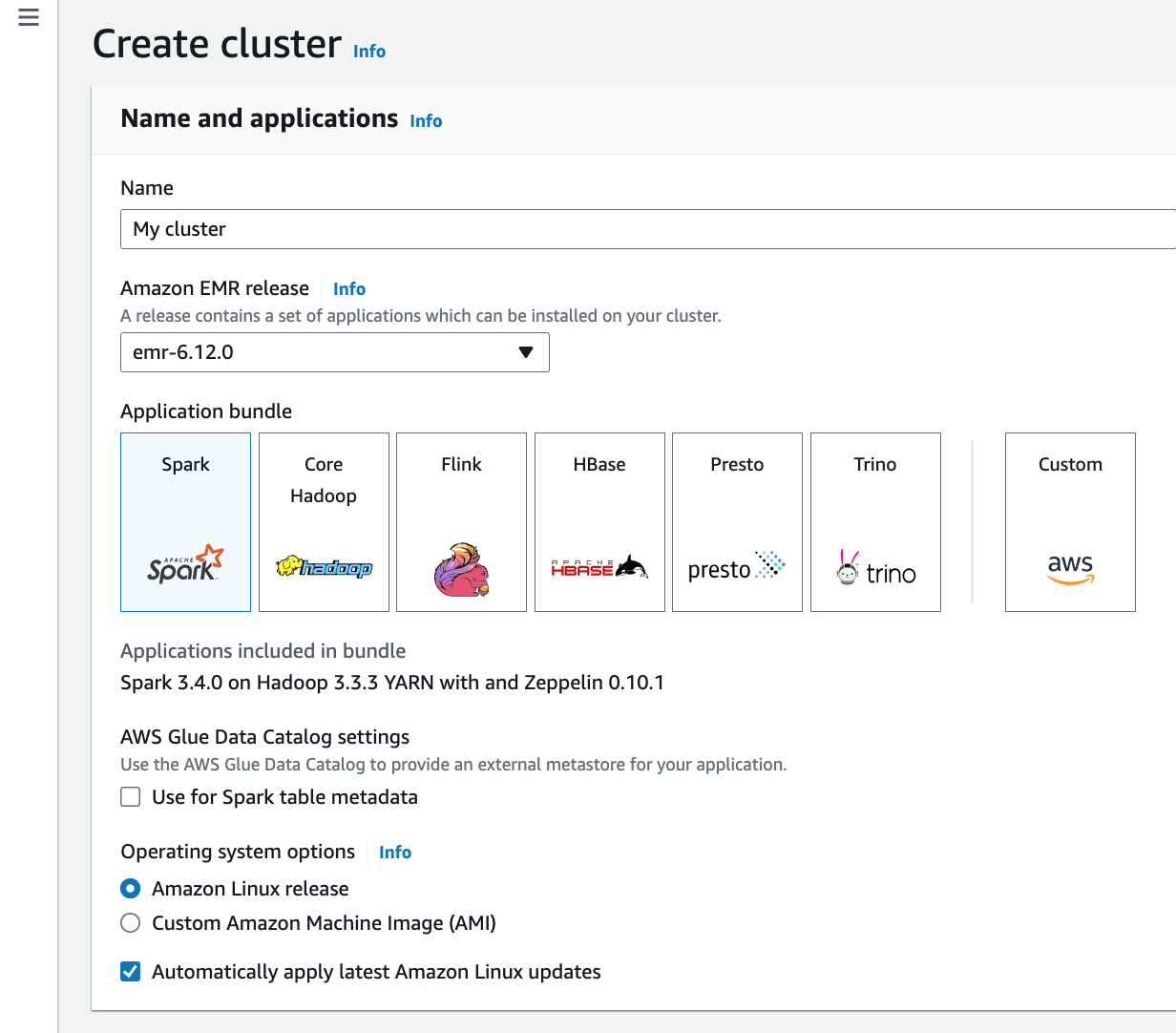
6) Do the same for spark2.py, spark2.py, spark2s.py, spark3.py, spark3s.py, spark4.py, spark4s.py

Note the demo file spark2s.py fails. Why?

Exercise 1)

Step A

Start up a EMR cluster as previously, but instead of choosing the “Core Hadoop” configuration chose the “Spark” configuration (see below), otherwise proceed as before.



Step B

Use the TestDataGen program from previous assignments to generate new data files.

Copy both generated files to the HDFS directory “/user/hadoop”

Step C

Load the ‘foodratings’ file as a ‘csv’ file into a DataFrame called foodratings. When doing so specify a schema having fields of the following names and types:

|  |  |
| --- | --- |
| Field Name | Field Type |
| name | String |
| food1 | Integer |
| food2 | Integer |
| food3 | Integer |
| food4 | Integer |
| placeid | Integer |

As the results of this exercise provide the magic number, *the code you execute* and screen shots of the following commands:

foodratings.printSchema()

foodratings.show(5)

Exercise 2)

Load the ‘foodplaces’ file as a ‘csv’ file into a DataFrame called foodplaces. When doing so specify a schema having fields of the following names and types:

|  |  |
| --- | --- |
| Field Name | Field Type |
| placeid | Integer |
| placename | String |

As the results of this exercise provide *the code you execute* and screen shots of the following commands:

foodplaces.printSchema()

foodplaces.show(5)

Exercise 3)

Step A

Register the DataFrames created in exercise 1 and 2 as tables called “foodratingsT” and “foodplacesT”

Step B

Use a SQL query on the table “foodratingsT” to create a new DataFrame called foodratings\_ex3a holding records which meet the following condition: food2 < 25 and food4 > 40. Remember, when defining conditions in your code use maximum parentheses.

As the results of this step *provide the code you execute* and screen shots of the following commands:

foodratings\_ex3a.printSchema()

foodratings\_ex3a.show(5)

Step C

Use a SQL query on the table “foodplacesT” to create a new DataFrame called foodplaces\_ex3b holding records which meet the following condition: placeid > 3

As the results of this step *provide the code you execute* and screen shots of the following commands:

foodplaces\_ex3b.printSchema()

foodplaces\_ex3b.show(5)

Exercise 4)

Use a transformation (not a SparkSQL query) on the DataFrame ‘foodratings’ created in exercise 1 to create a new DataFrame called foodratings\_ex4 that includes only those records (rows) where the ‘name’ field is “Mel” and food3 < 25.

As the results of this step provide the code you execute and screen shots of the following commands:

foodratings\_ex4.printSchema()

foodratings\_ex4.show(5)

Exercise 5)

Use a transformation (**not a SparkSQL query**) on the DataFrame ‘foodratings’ created in exercise 1 to create a new DataFrame called foodratings\_ex5 that includes only the columns (fields) ‘name’ and ‘placeid’

As the results of this step provide the code you execute and screen shots of the following commands:

foodratings\_ex5.printSchema()

foodratings\_ex5.show(5)

Exercise 6)

Use a transformation (**not a SparkSQL query**) to create a new DataFrame called ex6 which is the inner join, on placeid, of the DataFrames ‘foodratings’ and ‘foodplaces’ created in exercises 1 and 2

As the results of this step provide the code you execute and screen shots of the following commands:

ex6.printSchema()

ex6.show(5)