Apache Spark Exercise

Pre-Exercise Steps

1. Download the **spark\_workshop\_codebase.zip** file to the VM
2. Unzip the file. This will create a **spark\_workshop\_codebase** directory
3. Push the data into HDFS

ara

$ cd {path}/spark\_workshop\_codebase

$ hadoop fs -put spark\_data /user/cloudera/

1. Open the codebase in Eclipse
   1. Open Eclipse (Shortcut is on the VMs desktop)
   2. Click File -> Import
   3. Select Maven -> Existing Maven Projects and Click “Next >”
   4. Under Root Directory, click Browse
   5. Find the root directory of the project you unzipped. It should contain a pom.xml file. Then click Ok.
   6. Under Projects: you should see a tree structure loaded with excercise1, exercise2, exercise3 and more available.
   7. Click Finish

Running Spark Jobs (Exercise 1)

**Spark-Shell**

1. Open a Command Line Terminal
2. Start up a Spark-Shell

$ spark-shell --master yarn-client

1. Wait for the shell to come up with the following prompt

scala>

1. Type in the following Scala code

val list = List("who", "what", "when" ,"where", "how")

val data = sc.parallelize(list)

val wData = data.filter(\_.startsWith("w"))

wData.collect()

* 1. What did you just execute?
     1. The first two lines involve creating a simple list and then transforming that list into an RDD. This will take that data and distribute it to the Executors to be processed as soon as an action is performed.
     2. The third line is a transformation that takes the data within that RDD and filters it to only include those entries that contain a w in it. You will have noticed that this returns very quickly as no process has actually started to occur.
     3. The fourth and final line involves executing the action “collect” which returns all the values within the RDD after all the transformations have been applied.

1. After running the above code you should get the following result

res5: Array[String] = Array(who, what, when, where)

1. Congratulations you just ran a Spark job using Scala!

**Pyspark**

1. Start up a Pyspark Shell

$ pyspark --master yarn-client

1. Wait for the shell to up with the following prompt

>>>

1. Type in the following python code

list = ["who", "what", "when" ,"where", "how"]

data = sc.parallelize(list)

wData = data.filter(lambda x: x.startswith("w"))

wData.collect()

1. After running the above code you should get the result:

['who', 'what', 'when', 'where']

1. Congratulations you just ran a Spark job using Python

**Spark Submit**

1. Bring down the spark\_workshop\_codebase.zip file and extract it
2. Go to the spark\_workshop code base provided (on VM at /home/cloudera/spark-workshop/spark\_workshop\_codebase) and go to the exercise1 module. Run maven install to build the needed jar file:
   1. Note 1: The maven build has not been configured to set the main class. So when you submit the job you will need to define the main class to run as a command line argument.

$ cd {path}/spark\_workshop\_codebase/exercise1

$ mvn clean install

1. Verify the required jar was built

$ cd {path}/spark\_workshop\_codebase/exercise1/target

# The jar **com.clairvoyant.spark\_workshop.exercise1-jar-with-dependencies.jar** should have been built

1. Submit Java Code

$ cd {path}/spark\_workshop\_codebase/exercise1/target

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise1.java.Exercise1JavaSparkApp com.clairvoyant.spark\_workshop.exercise1-jar-with-dependencies.jar who what when where why how

#Output: [who, what, when, where, why]

1. Submit Scala Code

$ cd {path}/spark\_workshop\_codebase/exercise1/target

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise1.scala.Exercise1ScalaSparkApp com.clairvoyant.spark\_workshop.exercise1-jar-with-dependencies.jar who what when where why how

#Output: who what when where

1. Submit Python Code

$ cd {path}/

spark\_workshop\_codebase/exercise1/src/main/python

$ spark-submit Exercise1PythonSparkApp.py who what when were why how

#Output: ['who', 'what', 'when', 'where', 'why']

1. Congratulations you just ran a Spark job as a pre-packaged/built file!

Processing Access Log Data (Exercise 2)

1. In Eclipse, navigate to **excerise2** and select whichever programming language you would like (Java 8, Scala 11, or Python 2.7) and open the corresponding subdirectory and Main class.
2. Part of the exercise has been filled out to give you access to the initial DataFrame you need: accessLogDF
   1. Schema of accessLogDF:
      1. host -> String
      2. client\_indentd -> String
      3. user\_id -> String
      4. date\_time -> String
      5. method -> String
      6. endpoint -> String
      7. protocol -> String
      8. response\_code -> String
      9. content\_size -> String
      10. referrer -> String
      11. browser -> String
3. Below the DataFrame Definitions, you’ll find a couple TODO items. Complete each TODO.
   1. How to run each after your code is ready:
      1. Scala
         1. Right click on the project on the left side of Eclipse → Run AS → Maven Build.
         2. A dialog window opens, type: clean package for Goals and click on Run. This creates a jar file under the folder target.
         3. Go to the target folder from the command line. You will see the com.clairvoyant.spark\_workshop.exercise2-jar-with-dependencies.jar jar available.
         4. Execute the following:

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise2.scala.Exercise2ScalaSparkApp com.clairvoyant.spark\_workshop.exercise2-jar-with-dependencies.jar

* + 1. Python
       1. Go to the python folder under exercise 2 from the command line. You will see the Excercise2PythonSparkApp.py file available.
       2. Execute the following:

$ spark-submit Exercise2PythonSparkApp.py

* + 1. Java
       1. Right click on the project on the left side of Eclipse → Run AS → Maven Build.
       2. A dialog window opens, type: clean package for Goals and click on Run. This creates a jar file under the folder target.
       3. Go to the target folder from the command line. You will see the com.clairvoyant.spark\_workshop.exercise2-jar-with-dependencies.jar jar available.
       4. Execute the following:

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise2.java.Exercise2JavaSparkApp com.clairvoyant.spark\_workshop.exercise2-jar-with-dependencies.jar

* 1. Note: Possible answers will be listed a few pages bellow

Processing Company Data (Exercise 3) **(Optional)**

1. In Eclipse, navigate to **excerise3** and select whichever programming language you would like (Java 8, Scala 11, or Python 2.7) and open the corresponding subdirectory and Main class.
2. Part of the exercise has been filled out to give you access to the initial DataFrames you need: adddressDF, companyDF
   1. adddressDF Schema:
      1. address\_id -> String
      2. street\_address -> String
      3. city -> String
      4. state -> String
      5. zip\_code -> String
      6. country\_code -> String
   2. companyDF Schema:
      1. company\_id -> String
      2. company\_name -> String
      3. address\_id -> String
3. Below the DataFrame Definitions, you’ll find a couple TODO items. Complete each TODO.
   1. How to run each after your code is ready:
      1. Scala
         1. Right click on the project on the left side of Eclipse → Run AS → Maven Build.
         2. A dialog window opens, type: clean package for Goals and click on Run. This creates a jar file under the folder target.
         3. Go to the target folder from the command line. You will see the com.clairvoyant.spark\_workshop.exercise3-jar-with-dependencies.jar jar available.
         4. Execute the following:

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise3.scala.Exercise3ScalaSparkApp com.clairvoyant.spark\_workshop.exercise3-jar-with-dependencies.jar

* + 1. Python
       1. Go to the python folder under exercise 3 from the command line. You will see the Excercise3PythonSparkApp.py file available.
       2. Execute the following:

$ spark-submit --repositories http://repo1.maven.org/maven2 --packages com.databricks:spark-csv\_2.10:1.2.0 Exercise3PythonSparkApp.py

* + 1. Java
       1. Right click on the project on the left side of Eclipse → Run AS → Maven Build.
       2. A dialog window opens, type: clean package for Goals and click on Run. This creates a jar file under the folder target.
       3. Go to the target folder from the command line. You will see the com.clairvoyant.spark\_workshop.exercise3-jar-with-dependencies.jar jar available.
       4. Execute the following:

$ spark-submit --class com.clairvoyant.spark\_workshop.exercise3.java.Exercise3JavaSparkApp com.clairvoyant.spark\_workshop.exercise3-jar-with-dependencies.jar

* 1. Note: Possible answers will be listed a few pages bellow

Exercise 2 Possible Solutions

**Java**

1.

System.out.println("HealthCount: "+accesslogDF.filter("endpoint='/health'").count());

2. accesslogDF.filter("host='10.236.133.247'").write().saveAsTable("host\_entries");

**Python**

1.

print("Health Count: " + str(accesslogDF.filter("endpoint='/health'").count()))

2. accesslogDF.filter("host='10.236.133.247'").write.saveAsTable("host\_entries")

**Scala**

1.

println("Health count :"+accesslogDF.filter("endpoint='/health'").count())

2. accesslogDF.filter("host='10.236.133.247'").write().saveAsTable("host\_entries")

Exercise 3 Possible Solutions

**Java**

1.

sqlContext.sql("SELECT addresses.address\_id,addresses.street,addresses.city,addresses.state,addresses.zip\_code,addresses.country\_code,companies.company\_id,companies.name FROM addresses join companies on addresses.address\_id=companies.address\_id").write().mode("overwrite").saveAsTable("join\_company\_address");

2.

sqlContext.sql("SELECT \* FROM join\_company\_address WHERE state='AZ'").show();

**Python**

1.

sqlContext.sql("SELECT addresses.address\_id,addresses.street,addresses.city,addresses.state,addresses.zip\_code,addresses.country\_code,companies.company\_id,companies.name FROM addresses join companies on addresses.address\_id=companies.address\_id").write.mode("overwrite").saveAsTable("join\_company\_address")

2.

sqlContext.sql("SELECT \* FROM join\_company\_address WHERE state='AZ'").show()

**Scala**

1.

sqlContext.sql("SELECT addresses.address\_id,addresses.street,addresses.city,addresses.state,addresses.zip\_code,addresses.country\_code,companies.company\_id,companies.name FROM addresses join companies on addresses.address\_id=companies.address\_id").write.mode("overwrite").saveAsTable("join\_company\_address")

2.

sqlContext.sql("SELECT \* FROM join\_company\_address WHERE state='AZ'").show()