**Clairvoyant**

**Intro to Apache Spark Workshop:**

**Exercise Answers**

**Exercise 1 – Running Spark Jobs**

**Question:**

See Exercises Document

**Java Answer:**

N/A

**Scala Answer:**

N/A

**Python Answer:**

N/A

**Exercise 2 – Access Logs**

**Question:**

Analyze the access.log file and calculate the following:

* Count how many times the “/health” URL was hit.
* Map each line into the following tuple format (ip\_address, full\_line) and save the contents to HDFS.

Access log file can be found in two locations:

* In the spark-workshop-data.zip file provided, in the “logs” subdirectory.
* In HDFS (on the VM provided) at:

/user/cloudera/spark-workshop-data/logs/access.log

**Java Answer:**

JavaRDD<String> accessLogs = sc.textFile("/user/cloudera/spark-workshop-data/logs/access.log");  
JavaRDD<String> accessLogsHealth = accessLogs.filter(new Function<String, Boolean>() {  
 public Boolean call(String s) {  
 return s.contains("/health");  
 }  
});  
System.*out*.println(accessLogsHealth.count());  
//5470  
  
JavaRDD<Tuple2<String, String>> mappedAccessLogs = accessLogs.map(new Function<String, Tuple2<String, String>>() {  
 public Tuple2<String, String> call(String line) throws Exception {  
 String[] splitLine = line.split(" ");  
 return new Tuple2<String, String>(splitLine[0], line);  
 }  
});  
mappedAccessLogs.saveAsTextFile("/user/cloudera/spark-workshop-output-data/logs/access-logs-mapped");

**Scala Answer:**

**val** accessLogs = sc.textFile("/user/cloudera/spark-workshop-data/logs/access.log")  
accessLogs.filter(\_.contains("/health")).count()  
//res0: Long = 5470  
  
accessLogs.map(line => (line.split(" ")(0), line)).saveAsTextFile("/user/cloudera/spark-workshop-output-data/logs/access-logs-mapped")

**Python Answer:**

accessLogs = sc.textFile("/user/cloudera/spark-workshop-data/logs/access.log")  
accessLogs.filter(**lambda** x: "/health" **in** x).count()  
#5470  
  
accessLogs.map(**lambda** line: (line.split(" ")[0], line)).saveAsTextFile("/user/cloudera/spark-workshop-output-data/logs/access-logs-mapped")

**Exercise 3 – Joining Datasets**

**Question:**

Using the README.md and CHANGES.txt files, find out how many times the word “Spark” shows up in both of the files by joining the data together. Follow the bellow steps:

1. Create RDD’s for each file and filter each file to only keep all the instances of the work “Spark”
2. Perform a word count on each of the resulting datasets so the results are (K, V) pairs of type (word, count)
3. Join the two RDDs

Files can be found in two locations:

* In the spark-workshop-data.zip file provided, in the “spark” subdirectory
* In HDFS (on the VM provided) at

/user/cloudera/spark-workshop-data/spark/

**Java Answer:**

JavaRDD<String> readme = sc.textFile("/user/cloudera/spark-workshop-data/spark/README.md");  
JavaPairRDD<String, Integer> readmeWordCount = readme.flatMap(new FlatMapFunction<String, String>() {  
 public Iterable<String> call(String s) { return Arrays.*asList*(s.split(" ")); }  
}).filter(new Function<String, Boolean>() {  
 public Boolean call(String s) throws Exception {  
 return s.equals("Spark");  
 }  
}).mapToPair(new PairFunction<String, String, Integer>() {  
 public Tuple2<String, Integer> call(String s) {  
 return new Tuple2<String, Integer>(s, 1);  
 }  
}).reduceByKey(new Function2<Integer, Integer, Integer>() {  
 public Integer call(Integer a, Integer b) { return a + b; }  
});  
  
JavaRDD<String> changes = sc.textFile("/user/cloudera/spark-workshop-data/spark/CHANGES.txt");  
JavaPairRDD<String, Integer> changesWordCount = changes.flatMap(new FlatMapFunction<String, String>() {  
 public Iterable<String> call(String s) { return Arrays.*asList*(s.split(" ")); }  
}).filter(new Function<String, Boolean>() {  
 public Boolean call(String s) throws Exception {  
 return s.equals("Spark");  
 }  
}).mapToPair(new PairFunction<String, String, Integer>() {  
 public Tuple2<String, Integer> call(String s) {  
 return new Tuple2<String, Integer>(s, 1);  
 }  
}).reduceByKey(new Function2<Integer, Integer, Integer>() {  
 public Integer call(Integer a, Integer b) { return a + b; }  
});  
  
System.*out*.println(readmeWordCount.join(changesWordCount).collect());

**Scala Answer:**

**val** readme = sc.textFile("/user/cloudera/spark-workshop-data/spark/README.md")  
**val** readmeWordCount = readme.flatMap(line => line.split(" ")).filter(\_.equals("Spark")).map(word => (word, 1)).reduceByKey(\_ + \_)  
  
**val** changes = sc.textFile("/user/cloudera/spark-workshop-data/spark/CHANGES.txt")  
**val** changesWordCount = changes.flatMap(line => line.split(" ")).filter(\_.equals("Spark")).map(word => (word, 1)).reduceByKey(\_ + \_)  
  
readmeWordCount.join(changesWordCount).collect()  
//res0: Array[(String, (Int, Int))] = Array((Spark,(12,101)))

**Python Answer:**

readme = sc.textFile("/user/cloudera/spark-workshop-data/spark/README.md")  
readmeWordCount = readme.flatMap(**lambda** line: line.split(" ")).filter(**lambda** word: word == "Spark").map(**lambda** word: (word, 1)).reduceByKey(**lambda** a, b: a + b)  
  
changes = sc.textFile("/user/cloudera/spark-workshop-data/spark/CHANGES.txt")  
changesWordCount = changes.flatMap(**lambda** line: line.split(" ")).filter(**lambda** word: word == "Spark").map(**lambda** word: (word, 1)).reduceByKey(**lambda** a, b: a + b)  
  
readmeWordCount.join(changesWordCount).collect()  
# [(u'Spark', (12, 101))]

**Exercise 4 – Shared Variables**

**Question:**

In this exercise you will take a file with mock bank transaction data and process it using Shared Variables.

File can be found in two locations:

* In the spark-workshop-data.zip file provided, in the “transactions” subdirectory
* In HDFS (on the VM provided) at

/user/cloudera/spark-workshop-data/transactions/ user\_financial\_transactions.tsv

File is a tab-separated value file without a header. The file had the scheme:

UserID, Name, TransactionID, TransactionCode, Reason, BankID

Steps

1. Create a map with the following key value pairs (where the key is the TransactionCode and the value is a translated TransactionCode) and **Broadcast** it to the nodes:

C -> CASH\_ADVANCE

S -> BALANCE\_INQUIRY

B -> BALANCE\_TRANSFER

A -> OTHER

V -> OTHER

O -> OTHER

P -> PREAUTHORIZED

R -> AUTHORIZED

1. Use an **Accumulator** to count how many transactions from Bank “A” were of type “OTHER”.

**Java Answer:**

Map<String, String> transactionCodeMap = new HashMap<String, String>() {{  
 put("C", "CASH\_ADVANCE");  
 put("S", "BALANCE\_INQURIY");  
 put("B", "BALANCE\_TRANSFER");  
 put("A", "OTHER");  
 put("V", "OTHER");  
 put("O", "OTHER");  
 put("P", "PREAUTHORIZED");  
 put("R", "AUTHORIZED");  
}};  
final Broadcast transactionCodeMapBroadcast = sc.broadcast(transactionCodeMap);  
  
final Accumulator countAccum = sc.accumulator(0);  
  
JavaRDD transactionFile = sc.textFile("/user/cloudera/spark-workshop-data/transactions/user\_financial\_transactions.tsv");  
JavaRDD transactionData = transactionFile.map(new Function<String, String[]>() {  
 public String[] call(String line) throws Exception {  
 return line.split("\t");  
 }  
});  
transactionData = transactionData.filter(new Function<String[], Boolean>() {  
 public Boolean call(String[] line) throws Exception {  
 return line[5].equals("A");  
 }  
});  
  
transactionData.foreach(new VoidFunction<String[]>() {  
 public void call(String[] line) throws Exception {  
 Map<String, String> transactionCodeMap = (Map<String, String>) transactionCodeMapBroadcast.getValue();  
 if (transactionCodeMap.get(line[3]).equals("OTHER")) {  
 countAccum.add(1);  
 }  
 }  
});  
  
System.*out*.println(countAccum.value());  
//2

**Scala Answer:**

**val** transactionCodeMap = *Map*(  
 "C" -> "CASH\_ADVANCE",  
 "S" -> "BALANCE\_INQUIRY",  
 "B" -> "BALANCE\_TRANSFER",  
 "A" -> "OTHER",  
 "V" -> "OTHER",  
 "O" -> "OTHER",  
 "P" -> "PREAUTHORIZED",  
 "R" -> "AUTHORIZED"  
)  
**val** transactionCodeMapBroadcast = sc.broadcast(transactionCodeMap)  
  
**val** countAccum = sc.accumulator(0)  
  
**case class** Transaction (userId: String, name: String, transactionId: String, transactionCode: String, reason: String, bankId: String)  
  
**val** transactionFile = sc.textFile("/user/cloudera/spark-workshop-data/transactions/user\_financial\_transactions.tsv")  
**val** transactionData = transactionFile.map(\_.split("\t")).map(  
 r => *Transaction*(r(0), r(1), r(2), r(3), r(4), r(5))  
)  
transactionData.filter(r => r.bankId.equals("A")).foreach(  
 line => **if**(transactionCodeMapBroadcast.value(line.transactionCode).equals("OTHER")) {  
 countAccum += 1  
 }  
)

countAccum.value  
//res1: Int = 2

**Python Answer:**

transactionCodeMap = {  
 "C": "CASH\_ADVANCE",  
 "S": "BALANCE\_INQUIRY",  
 "B": "BALANCE\_TRANSFER",  
 "A": "OTHER",  
 "V": "OTHER",  
 "O": "OTHER",  
 "P": "PREAUTHORIZED",  
 "R": "AUTHORIZED"  
}  
transactionCodeMapBroadcast = sc.broadcast(transactionCodeMap)  
  
countAccum = sc.accumulator(0)  
  
transactionFile = sc.textFile("/user/cloudera/spark-workshop-data/transactions/user\_financial\_transactions.tsv")  
  
**def accumFunction**(line):  
 **global** countAccum  
 **if** transactionCodeMapBroadcast.value[line[3]] == "OTHER":  
 countAccum += 1  
  
transactionData = transactionFile.map(**lambda** line: line.split("\t"))  
transactionData.filter(**lambda** line: line[5] == "A").foreach(accumFunction)  
  
countAccum.value  
#2