**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 company.tsv and address.tsv files, join the names data with the address data.

Data Scheme

* company.tsv
  + company\_id, company\_name, address\_id
* address.tsv
  + address\_id, street\_address, city, state, zip\_code

Files can be found in two locations:

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

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

**Java Answer:**

final int companyAddressIDColumnIndex = 2;  
final int addressAddressIDColumnIndex = 0;  
  
JavaRDD<String> company = sc.textFile("/user/cloudera/spark-workshop-data/join/company.tsv");  
JavaRDD<String> address = sc.textFile("/user/cloudera/spark-workshop-data/join/address.tsv");  
  
JavaPairRDD<String, String> companyParsed = company.mapToPair(new PairFunction<String, String, String>() {  
 public Tuple2<String, String> call(String line) throws Exception {  
 String[] lineSplit = line.split("\t");  
 return new Tuple2<String, String>(lineSplit[companyAddressIDColumnIndex], line);  
 }  
});  
JavaPairRDD<String, String> addressParsed = address.mapToPair(new PairFunction<String, String, String>() {  
 public Tuple2<String, String> call(String line) throws Exception {  
 String[] lineSplit = line.split("\t");  
 return new Tuple2<String, String>(lineSplit[addressAddressIDColumnIndex], line);  
 }  
});  
  
List<Tuple2<String, Tuple2<String, String>>> list = companyParsed.join(addressParsed).collect();  
for(Tuple2<String, Tuple2<String, String>> entry : list) {  
 System.*out*.println(entry);  
}

// (A010,(C010 Cloudera A010,A010 1001 Page Mill Rd Palo Alto CA 94305))  
// (A001,(C001 Clairvoyant A001,A001 6185 W Detroit St Chandler AZ 85226))  
// (A001,(C003 Bluecanary A001,A001 6185 W Detroit St Chandler AZ 85226))  
// (A007,(C007 Shutterfly A007,A007 6997 S Shutterfly Way Tempe AZ 85283))  
// (A005,(C006 Grand Canyon University A005,A005 3300 W Camelback Rd Phoenix AZ 85017))  
// (A003,(C004 Apollo Education Group A003,A003 4025 S Riverpoint Pkwy Phoenix AZ 85040))  
// (A009,(C009 P.F. Chang's A009,A009 7676 E. Pinnacle Peak Rd. Scottsdale AZ 85255))  
// (A002,(C002 Intel A002,A002 5000 W Chandler Blvd Chandler AZ 85226))  
// (A004,(C005 Early Warning A004,A004 6552 N 90th St #100 Scottsdale AZ 85260))  
// (A011,(C011 Amazon A011,A011 410 Terry Ave. North Seattle WA 98109))  
// (A008,(C008 Altisource A008,A008 2501 TX-121 BUS #300 Lewisville TX 75067))

**Scala Answer:**

**val** companyAddressIDColumnIndex = 2  
 **val** addressAddressIDColumnIndex = 0  
  
 **val** company = sc.textFile("/user/cloudera/spark-workshop-data/join/company.tsv")  
 **val** address = sc.textFile("/user/cloudera/spark-workshop-data/join/address.tsv")  
  
 **val** companyParsed = company.map( line => (line.split("\t")(companyAddressIDColumnIndex), line))  
 **val** addressParsed = address.map( line => (line.split("\t")(addressAddressIDColumnIndex), line))  
  
 companyParsed.join(addressParsed).collect().foreach(*println*)  
// (A010,(C010 Cloudera A010,A010 1001 Page Mill Rd Palo Alto CA 94305))  
// (A001,(C001 Clairvoyant A001,A001 6185 W Detroit St Chandler AZ 85226))  
// (A001,(C003 Bluecanary A001,A001 6185 W Detroit St Chandler AZ 85226))  
// (A007,(C007 Shutterfly A007,A007 6997 S Shutterfly Way Tempe AZ 85283))  
// (A005,(C006 Grand Canyon University A005,A005 3300 W Camelback Rd Phoenix AZ 85017))  
// (A003,(C004 Apollo Education Group A003,A003 4025 S Riverpoint Pkwy Phoenix AZ 85040))  
// (A009,(C009 P.F. Chang's A009,A009 7676 E. Pinnacle Peak Rd. Scottsdale AZ 85255))  
// (A002,(C002 Intel A002,A002 5000 W Chandler Blvd Chandler AZ 85226))  
// (A004,(C005 Early Warning A004,A004 6552 N 90th St #100 Scottsdale AZ 85260))  
// (A011,(C011 Amazon A011,A011 410 Terry Ave. North Seattle WA 98109))  
// (A008,(C008 Altisource A008,A008 2501 TX-121 BUS #300 Lewisville TX 75067))

**Python Answer:**

companyAddressIDColumnIndex = 2  
 addressAddressIDColumnIndex = 0  
  
 company = sc.textFile("/user/cloudera/spark-workshop-data/join/company.tsv")  
 address = sc.textFile("/user/cloudera/spark-workshop-data/join/address.tsv")  
  
 companyParsed = company.map(**lambda** line: (line.split("\t")[companyAddressIDColumnIndex], line))  
 addressParsed = address.map(**lambda** line: (line.split("\t")[addressAddressIDColumnIndex], line))  
  
 list = companyParsed.join(addressParsed).collect()  
 **for** entry **in** list:  
 **print**(entry)  
# (u'A009', (u"C009\tP.F. Chang's\tA009", u'A009\t7676 E. Pinnacle Peak Rd.\tScottsdale\tAZ\t85255'))  
# (u'A007', (u'C007\tShutterfly\tA007', u'A007\t6997 S Shutterfly Way\tTempe\tAZ\t85283'))  
# (u'A010', (u'C010\tCloudera\tA010', u'A010\t1001 Page Mill Rd\tPalo Alto\tCA\t94305'))  
# (u'A005', (u'C006\tGrand Canyon University\tA005', u'A005\t3300 W Camelback Rd\tPhoenix\tAZ\t85017'))  
# (u'A003', (u'C004\tApollo Education Group\tA003', u'A003\t4025 S Riverpoint Pkwy\tPhoenix\tAZ\t85040'))  
# (u'A001', (u'C001\tClairvoyant\tA001', u'A001\t6185 W Detroit St \tChandler\tAZ\t85226'))  
# (u'A001', (u'C003\tBluecanary\tA001', u'A001\t6185 W Detroit St \tChandler\tAZ\t85226'))  
# (u'A011', (u'C011\tAmazon\tA011', u'A011\t410 Terry Ave. North\tSeattle\tWA\t98109'))  
# (u'A004', (u'C005\tEarly Warning\tA004', u'A004\t6552 N 90th St #100\tScottsdale\tAZ\t85260'))  
# (u'A002', (u'C002\tIntel\tA002', u'A002\t5000 W Chandler Blvd\tChandler\tAZ\t85226'))  
# (u'A008', (u'C008\tAltisource\tA008', u'A008\t2501 TX-121 BUS #300 \tLewisville\tTX 75067'))

**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