Assignment # 3

After you finish Lab 5, and while still in spark-shell in the VM, use the mydata RDD and Scala Spark wordcount example in class to perform word count on the frostroad.txt file. But instead of writing the output to HDFS or calling .collect(), use .foreach(println) at the end of the program to display all the output in a column.

Ans:

```
val myData = sc.textFile("file:/home/training/training_materials/data/frostroad.txt")
18/02/18 08:38:55 INFO storage.MemoryStore: ensureFreeSpace(280171) called with
curMem=0, maxMem=280248975
18/02/18 08:38:55 INFO storage.MemoryStore: Block broadcast_0 stored as values in
memory (estimated size 273.6 KB, free 267.0 MB)scala> val counts =
myData.flatMap(line => line.split("\\W+")).map(word =>
(word.toLowerCase,1)).reduceByKey(_ + _)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[21] at reduceByKey at
<console>:23

scala> val sortedCount = counts.sortByKey().count()

sortedCount: Long = 95

scala> val sorted = counts.sortByKey().foreach(println)

(,3)
(a,3)
(about,1)
(ages,2)
(all,1)
```

Assignment 4:

Use RDDs

1. Make a new HDFS directory called "/loudacre/weblogs".
2. Move all of the files in the Linux directory /home/training/training_materials/data/weblogs/ to the HDFS directory created in step 1.
3. Create a val "logfiles" that refers to the weblogs HDFS directory and use this in subsequent commands to save some typing.
4. Create a val "input" that is an RDD containing all of the records in the logfiles directory.
5. Create a val "inputJPG" that contains only the records from input that contain jpg requests
6. View the first 10 records using take.
7. Combine the functions of steps 4 and 5 in one line of code, adding a count of the records too. What is that count?
8. Use the map function to get the length of each record, and show the first 10 results.
9. Use the map function, splitting on space, to show the individual fields in each record, and show the first 10 results.
10. Use the previous results to show just the IP addresses in each records and show the first 10 results.
11. Use the .foreach(println) technique to make the results in step 10 easier to read.
12. Save the results from step 10 in the HDFS file "/loudcare/iplist".
13. Use the -ls and -cat command options (and optionally the Hue browser) to show the HDFS

directory “/loudcare/iplist” as well as the contents of the part file containing the IP addresses.
14. Write a program to display just the IP addresses and timestamps for all the records that contain jpg requests in the format “IPAddress/Timestamp” and show the first 10 results.
15. Turn in the lines of code you used

Results of Lab 6:

1. hdfs dfs -mkdir /loudacre/weblogs
2. hdfs dfs -copyFromLocal /home/training/training_materials/data/weblogs/* /loudacre/weblogs
3. val logfiles = "hdfs:///loudacre/weblogs/"
logfiles: String = hdfs:///loudacre/weblogs/
4.scala> val input = sc.textFile(logfiles)
18/02/24 21:41:30 INFO storage.MemoryStore: ensureFreeSpace(280171) called with curMem=0, maxMem=280248975
5. scala> val inputJPG = input.filter(x => x.contains("jpg"))
inputJPG: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:25
6. scala> inputJPG.take(10)
"GET /ronin_s4.jpg HTTP/1.0" 200 5552 "http://www.loudacre.com" "Loudacre Mobile Browser MeeToo 1.0", 104.184.210.93
- 28402 [15/Sep/2013:23:42:53 +0100] "GET /titanic_2200.jpg HTTP/1.0" 200 19466 "http://www.loudacre.com"
"Loudacre Mobile Browser MeeToo 2.0", 37.91.137.134 - 36171
[15/Sep/2013:23:39:33 +0100] "GET /ronin_novelty_note_3.jpg
HTTP/1.0" 200 7432 "http://www.loudacre.com" "Loudacre Mobile Browser iFruit 3", 177.43.223.203 - 90653
[15/Sep/2013:23:31:17 +0100] "GET /ifruit_3.jpg HTTP/1.0" 200 19578 "http://www.loudacre.com"
"Loudacre Mobile Browser Sorrento F31L", 19.250.65.76 - 44388
[15/Sep/2013:23:31:10 +0100]
"GET /sorrento_f24l.jpg HTTP/1.0" 200 5730 "http://www.loudacre.com" "Loudacre M...

7.scala> val counts= input.filter(x => x.contains("jpg")).count()
counts: Long = 64978
8. inputJPG.map(x=> x.length).take(10)
res1: Array[Int] = Array(153, 158, 162, 157, 155, 155, 156, 156, 166, 156)
9. scala> val fieldSplit = input.map(x=> x.split(" ")).take(10)
fieldSplit: Array[Array[String]] = Array(Array(3.94.78.5, -, 69827, [15/Sep/2013:23:58:36, +0100],
"GET, /KBDOC-00033.html, HTTP/1.0", 200, 14417, "http://www.loudacre.com", "", "Loudacre, Mobile, Browser, iFruit, 1"), Array(3.94.78.5, -, 69827,

10. scala> val ipAddressMap = input.map(x=> (x.split(" ")(0), 1))
ipAddressMap: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:25

scala> val ipAddress = ipAddressMap.take(10)

ipAddress: Array[(String, Int)] = Array((3.94.78.5,1), (3.94.78.5,1), (19.38.140.62,1),
(19.38.140.62,1),
(129.133.56.105,1), (129.133.56.105,1), (217.150.149.167,1), (217.150.149.167,1),
(217.150.149.167,1),
(217.150.149.167,1))
11. scala> ipAddress.foreach(println)
(3.94.78.5,1)
(3.94.78.5,1)

18/02/24 22:01:50 INFO storage.BlockManager: Removing broadcast 1
13.[training@localhost ~]$ hdfs dfs -ls /loudacre/iplist
Found 183 items
-rw-rw-rw- 1 training supergroup 0 2018-02-24 22:03 /loudacre/iplist/_SUCCESS
-rw-rw-rw- 1 training supergroup 63025 2018-02-24 22:03 /loudacre/iplist/part-00000

[training@localhost ~]$ hdfs dfs -cat /loudacre/iplist/part-00000
(3.94.78.5,1)
(3.94.78.5,1)
(19.38.140.62,1)
(19.38.140.62,1)
(129.133.56.105,1)
14. scala> val ipAddressMapWithTimeStamp = input.filter(record =>
record.contains("jpg")).map(x => x.split(" ")).map(x => x(0) + "/" + x(3))
ipAddressMapWithTimeStamp: org.apache.spark.rdd.RDD[String] =
MapPartitionsRDD[4] at map at <console>:25

scala> ipAddressMapWithTimeStamp.take(10).foreach(println)
217.150.149.167/[15/Sep/2013:23:56:06
104.184.210.93/[15/Sep/2013:23:42:53
37.91.137.134/[15/Sep/2013:23:39:33

Assignment 5:

Output for Lab 7:
scala> val userSchema = sc.textFile("/loudacre/weblogs/*6.log")
18/03/02 16:22:00 INFO storage.MemoryStore: ensureFreeSpace(280171) called with
curMem=0, maxMem=280248975

1. a. scala> val userIdPairRdd = userSchema.map(_.split(" ")).map(x => (x(2), 1))
userIdPairRdd: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[7] at map at
<console>:23

scala> userIdPairRdd.take(4).foreach(println)
(96828,1)

(96828,1)
(58687,1)
(58687,1)

b. scala> val userIdSum = userIdPairRdd.reduceByKey(_+_)
userIdSum: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[39] at reduceByKey at <console>:25
scal> userIdSum.take(4).foreach(println)
(99066,4)
(27120,8)
(72537,4)
(17943,2)

2. a. scala> val swapKeyValue = userIdSum.map(_.swap)
swapKeyValue: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[9] at map at <console>:27

scala> swapKeyValue.take(4).foreach(println)
(4,99066)
(8,27120)
(4,72537)
(2,17943)
scala> val countByFreq = swapKeyValue.countByKey()
countByFreq: scala.collection.Map[Int,Long] = Map(138 -> 10, 5 -> 30, 120 -> 2, 10 -> 834, 142 -> 5, 24 -> 6, 14 -> 317, 20 -> 37, 152 -> 6, 164 -> 3, 106 -> 2, 132 -> 7, 116 -> 7, 6 -> 2064, 28
3. scala> val userIdIpPairRdd = userSchema.map(_.split(" ")).map(x => (x(2), x(0))).groupByKey()
userIdIpPairRdd: org.apache.spark.rdd.RDD[(String, Iterable[String])] = ShuffledRDD[42] at groupByKey at <console>:23
scala> userIdIpPairRdd.take(5).foreach(println)
(99066,CompactBuffer(77.122.77.207, 77.122.77.207, 1.87.81.30, 1.87.81.30))
(27120,CompactBuffer(164.96.191.89, 164.96.191.89, 130.35.60.234, 130.35.60.234,

4. scala> val accountSchema = sc.textFile("/loudacre/accounts")
a. scala> accountSchema.take(2).foreach(println)
1,2008-10-23 16:05:05.0,\N,Donald,Becton,2275 Washburn

scala> val accountRdd = accountSchema.map(x => (x.split(",")(0), x))
accountRdd: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[44] at map at <console>:23
scala> accountRdd.take(5)
(1,1,2008-10-23 16:05:05.0,\N,Donald,Becton,2275 Washburn Street,Oakland,CA,94660,5100032418,2014-03-18 13:29:47.0,2014-03-18 13:29:47.0)

b. scala> val joinUserIdHitCountWithAccount = accountRdd.join(userIdSum)

joinUserIdHitCountWithAccount: org.apache.spark.rdd.RDD[(String, (String, Int))] = MapPartitionsRDD[47] at join at <console>:31
scala> joinUserIdHitCountWithAccount.take(5).foreach(println)

(27120,(27120,2011-12-01 09:59:17.0,2013-09-07 08:29:37.0,Dale,Hanson,2578 Ingram Road,Inglewood,CA,90309,3107913634,2014-03-18 13:30:36.0,2014-03-18 13:30:36.0,8))
(92694,(92694,2013-01-25 04:11:10.0,2013-12-04 02:31:35.0,Stacy,Allbritton,4990 Clearview

c. scala> val filteredSet = joinUserIdHitCountWithAccount.map(x => (x._2._1.split(",")(0), x._2._2, x._2._1.split(",")(3),x._2._1.split(",")(4))
filteredSet: org.apache.spark.rdd.RDD[(String, Int, String, String)] = MapPartitionsRDD[38] at map at <console>:33

scala> filteredSet.take(5).foreach(println)
(27120,8,Dale,Hanson)
(92694,2,Stacy,Allbritton)
(40581,2,Norman,Scanlon)
(25665,4,Ann,McElroy)
(105237,4,Kathryn,McCallum)

Assignment 6:

The cluster HDFS directory /SEIS736/tweets contains one file of all the tweets from about a 1-year period that contain keywords potentially related to GPS. The schema for each line is:Screen name, Tweet, Language, Timestamp, GeoLocation

These fields are all strings in a tab-separated line. For this assignment, you only need the Tweet field. Note that there are a variety of errors in the data.

Your assignment is to write a Scala/Spark program using spark-shell on the cluster, to produce a list of the top words (as the keys) used in these tweets with their corresponding word counts (as the values), in decreasing count order. Words must start with a letter and need to be at least 3 characters long. Turn in your source code and the top 100 most frequent words appearing in the tweets.

val gpsTweetFile = sc.textFile("/SEIS736/tweets")
gpsTweetFile: org.apache.spark.rdd.RDD[String] = /SEIS736/tweets MapPartitionsRDD[1] at textFile at <console>:27

scala> val tweetsList = gpsTweetFile.map(x => x.split("\\t")).collect{case x =>if (x.length>1) x(1)}.map(x=>x.toString)
tweetsList: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[5] at map at <console>:29

scala> tweetsList.take(5)
res1: Array[String] = Array(Are you a recent computer science graduate (or similar course) looking to start your career with a global software house? Get in touch ASAP, RT @DarrenDalrymple: Are you a recent computer science graduate (or similar course) looking to start your career with a global software hou…, RT @DarrenDalrymple: Are you a recent

computer science graduate (or similar course) looking to start your career with a global software hou…, RT @DarrenDalrymple: Are you a recent computer science graduate (or similar course) looking to start your career with a global software hou…, RT @DarrenDalrymple: Are you a recent computer science graduate (or similar course) looking to start your career with a global software hou…)

```
scala> val filteredTweets =tweetsList.flatMap(x=>x.split(" ")).filter(x => x.matches("[A-Za-z]+") && x.length >=3)
filteredTweets: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[7] at filter at <console>:31
scala> filteredTweets.take(2)
res2: Array[String] = Array(Are, you)
scala> val filteredTweetsCount= filteredTweets.map(x=>(x.toLowerCase,1)).reduceByKey(_+_)
filteredTweetsCount: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[9] at reduceByKey at <console>:33
scala> val filteredTweetsCount= filteredTweets.map(x=>(x.toLowerCase,1)).reduceByKey(_+_)
filteredTweetsCount: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[28] at reduceByKey at <console>:33
scala> val sortedTweetsByCount = filteredTweetsCount.sortBy(_._2, false)
sortedTweetsByCount: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[48] at sortBy at <console>:35
scala> sortedTweetsByCount.take(100).foreach(println)
(your,11659)
(power,8464)
(career,6933)
```

Assignment 9:

I have a log file from a tool called iStumbler on the cluster in a directory called /SEIS736/WiFi. Write a Scala/Spark program, using Eclipse on the VM to create a .jar file to be uploaded to the cluster, to look at the “WiFiScan” records that produces a list of all the different security protocols present (for example, “WEP”), along with a count of how many unique wireless access points use each protocol. Note that the data contains multiple records for the same MAC address, so your program must ensure that a security protocol is only counted once for each unique MAC address (unique wireless access point). Your solution should not use Dataframes or SparkSQL. Your output should go into one part file in the HDFS output directory.

Pick reasonable values for the spark-submit command. Turn in your Spark .scala program, the spark-submit command you used, and your output.

```
import org.apache.spark.{SparkConf, SparkContext}

object IStumblerWiFi {

def main(args: Array[String]) {

  if (args.length < 3) {
```

```
    println("Usage: IStumblerWiFi <Input> <output> <NumOutputFiles>")
    System.exit(1)
    }
  val sparkConf = new SparkConf().setAppName("IStumblerWiFi ")
  val sc = new SparkContext(sparkConf)
 sc.textFile(args(0))
  .filter(x=>x.contains("WiFiScan"))
  .mapPartitionsWithIndex { (index, it) => if (index == 0) it.drop(1) else it }
  .map(x=>x.split(",")).keyBy(x=>(x(4)))
  .mapValues(x=>(x(2))).distinct.map(x=>(x._1,1))
  .reduceByKey(_+_,1)
  .map{case (k,v)=>(v,k)}
  .sortByKey(ascending = false)
  .saveAsTextFile(args(1))
  System.exit(0)
```

Sbt file:

SparkWifi.sbt:

```
name := "Spark Wifi Project"
version := "1.0"
scalaVersion := "2.11.6"
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.3.0"
```

Moving jar to cluster: scp ~/test/target/scala-2.11/spark-wifi-project_2.11-1.0.jar khan7912@hc.gps.stthomas.edu:/home/khan7912

```
hadoop fs -rm -r IStumblerWiFiOut
spark-submit \ --class IStumblerWiFi \
      --master yarn-cluster \
      /home/khan7912/spark-wifi-project_2.11-1.0.jar\
      /SEIS736/WiFi/ \
```

IStumblerWiFiOut\. 1

```
hadoop fs -cat IStumblerWiFiOut/part-00000
[khan7912@hc ~]$ hadoop fs -cat iStumblerWiFiOut/part-00000
(231,WPA 2)
(43,Open)
(5,Enterprise WPA 2)
(5,WPA)
(1,WEP)
```