

Case Study 5

1. There are two parts this case study

▪ First Part

You have to create a Spark Application which streams data from a file on local directory on your machine and does the word count on the fly. The word should be done by the spark application in such a way that as soon as you drop the file in your local directory, your spark application should immediately do the word count for you.

Step 1: Program for word count

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.rdd.RDD
import org.apache.spark.streaming.{Seconds, StreamingContext, Time}
import org.apache.spark.sql.SparkSession
import org.apache.log4j.{Level, Logger}

object SqlNetworkWordCount {

  def main(args: Array[String]): Unit = {
    println("hey Spark SQL Streaming")
    val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
    val sc = new SparkContext(conf)
    val rootLogger =Logger.getRootLogger()
    rootLogger.setLevel(Level.ERROR)
    println("hey Spark Streaming ---> 1")
    //val sparkConf = new SparkConf().setAppName("NetworkWordCount")
```

```

println("hey Spark Streaming ---> 2")
val ssc = new StreamingContext(sc, Seconds(10))
val lines = ssc.socketTextStream("localhost", 9999)
println("hey Spark Streaming ---> 3")
val words = lines.flatMap(_.split(" "))

// Convert RDDs of the words DStream to DataFrame and run SQL query
words.foreachRDD { (rdd: RDD[String], time: Time) =>
    val spark = SparkSessionSingleton.getInstance(rdd.sparkContext.getConf)
    import spark.implicits._

    // Convert RDD[String] to RDD[case class] to DataFrame
    val wordsDataFrame = rdd.map(w => Record(w)).toDF()

    // Creates a temporary view using the DataFrame
    wordsDataFrame.createOrReplaceTempView("words")

    // Do word count on table using SQL and print it
    val wordCountsDataFrame =
        spark.sql("select word, count(*) as total from words group by word")
    println(s"===== $time =====")
    wordCountsDataFrame.show()
}

ssc.start()
ssc.awaitTermination()
}

/** Case class for converting RDD to DataFrame */

```

```
case class Record(word: String)

/** Lazily instantiated singleton instance of SparkSession */
object SparkSessionSingleton {

  @transient private var instance: SparkSession = _

  def getInstance(sparkConf: SparkConf): SparkSession = {

    if (instance == null) {

      instance = SparkSession

        .builder

        .config(sparkConf)

        .getOrCreate()

    }

    instance

  } } }
```

```
SqlNetworkWordCount.scala 8
1
2+ import org.apache.spark.{SparkConf, SparkContext}
7
8= object SqlNetworkWordCount {
9
10= def main(args: Array[String]): Unit = {
11     println("hey Spark SQL Streaming");
12
13     val conf = new SparkConf().setMaster("local[2]").setApp
14     val sc = new SparkContext(conf);
15     val rootLogger = Logger.getRootLogger();
16     rootLogger.setLevel(Level.ERROR);
17
18
19     println("hey Spark Streaming ---> 1");
20     //val sparkConf = new SparkConf().setAppName("NetworkW
21     println("hey Spark Streaming ---> 2");
22     val ssc = new StreamingContext(sc, Seconds(10));
23     val lines = ssc.socketTextStream("localhost", 9999);
24     println("hey Spark Streaming ---> 3");
```

```
SqlNetworkWordCount.scala 8
25
26     val words = lines.flatMap(_.split(" "));
27
28     // Convert RDDs of the words DStream to DataFrame and
29     words.foreachRDD { (rdd: RDD[String], time: Time) =>
30         val spark = SparkSessionSingleton.getInstance(rdd.sp
31         import spark.implicits._;
32
33         // Convert RDD[String] to RDD[case class] to DataFrame
34         val wordsDataFrame = rdd.map(w => Record(w)).toDF();
35
36         // Creates a temporary view using the DataFrame
37         wordsDataFrame.createOrReplaceTempView("words");
38
39         // Do word count on table using SQL and print it
40         val wordCountsDataFrame =
41             spark.sql("select word, count(*) as total from wor
42         println(s"===== $time =====")
43         wordCountsDataFrame.show()
44     }
45
```

```
SqlNetworkWordCount.scala  ✖
46
47   ssc.start()
48   ssc.awaitTermination()
49
50 }
51
52
53 /** Case class for converting RDD to DataFrame */
54 case class Record(word: String)
55
56 /** Lazily instantiated singleton instance of SparkSession */
57 object SparkSessionSingleton {
58
59   @transient private var instance: SparkSession = _
60
61   def getInstance(sparkConf: SparkConf): SparkSession = {
62     if (instance == null) {
63       instance = SparkSession
64         .builder
65         .config(sparkConf)
```

```
SqlNetworkWordCount.scala  ✖
55
56 /** Lazily instantiated singleton instance of SparkSession */
57 object SparkSessionSingleton {
58
59   @transient private var instance: SparkSession = _
60
61   def getInstance(sparkConf: SparkConf): SparkSession = {
62     if (instance == null) {
63       instance = SparkSession
64         .builder
65         .config(sparkConf)
66         .getOrCreate()
67     }
68     instance
69   }
70 }
71
72 }
73
74
```

Step 2: Start netcat from a terminal

nc -lk 9999

```
acadgild@localhost:~  
[acadgild@localhost ~]$ nc -lk 9999  
word  
Hey Acadgild  
wordcount wordcount wordcount  
Bigdata Bigdata Bigdata Bigdata
```

Step 3: Display Results:

word = 1

Hey =1

Acadgild = 1

Wordcount = 3

Bigdata = 4

```
Problems Tasks Console  
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Ap  
===== 1525072840000 ms =====  
+-----+  
|word|total|  
+-----+  
|word|    1|  
+-----+  
  
===== 1525072850000 ms =====  
+-----+  
|    word|total|  
+-----+  
|Acadgild|    1|  
|    Hey|    1|  
+-----+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Apr 30, 2016 10:00:00 AM)
+-----+
===== 1525072890000 ms =====
+-----+
|word|total|
+-----+
+-----+

===== 1525072900000 ms =====
+-----+
|    word|total|
+-----+
|wordcount|    3|
+-----+
```

```
Problems Tasks Console
SqlNetworkWordCount$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Apr 30, 2016 10:00:00 AM)
+-----+
===== 1525072940000 ms =====
+-----+
|word|total|
+-----+
+-----+

===== 1525072950000 ms =====
+-----+
|    word|total|
+-----+
|Bigdata|    4|
+-----+
```

▪ **Second Part - In this part, you will have to create a Spark Application which should do the following**

- 1. Pick up a file from the local directory and do the word count**
- 2. Then in the same Spark Application, write the code to put the same file on HDFS.**
- 3. Then in same Spark Application, do the word count of the file copied on HDFS in step 2**
- 4. Lastly, compare the word count of step 1 and 2. Both should match, other throw an error**

Step 1: Program of Spark HDFS WordCount Comparison

```
import java.io.File

import org.apache.spark.{SparkConf, SparkContext}
import scala.io.Source._
import org.apache.log4j.{Level, Logger}

object SparkHDFSWordCountComparison {

  private var localFilePath: File = new File("/home/acadgild/test.txt")

  private var dfsDirPath: String = "hdfs://localhost:8020/user/streaming"

  private val NPARAMS = 2

  def main(args: Array[String]): Unit = {

    //parseArgs(args)

    println("SparkHDFSWordCountComparison : Main Called Successfully")

    println("Performing local word count")

    val fileContents = readFile(localFilePath.toString())

    println("Performing local word count - File Content ->>" + fileContents)

    val localWordCount = runLocalWordCount(fileContents)
```



```
println("SparkHDFSWordCountComparison : Main Called Successfully ->
Local Word Count is ->>" + localWordCount)

println("Performing local word count Completed !!")

println("Creating Spark Context")

val conf = new
SparkConf().setMaster("local[2]").setAppName("SparkHDFSWordCountComp
arisonApp")

    val sc = new SparkContext(conf)
    val rootLogger = Logger.getRootLogger()
    rootLogger.setLevel(Level.ERROR)
    println("Spark Context Created")
    println("Writing local file to DFS")
    val dfsFilename = dfsDirPath + "/dfs_read_write_test"
    val fileRDD = sc.parallelize(fileContents)
    fileRDD.saveAsTextFile(dfsFilename)
    println("Writing local file to DFS Completed")
    println("Reading file from DFS and running Word Count")
    val readFileRDD = sc.textFile(dfsFilename)
    val dfsWordCount = readFileRDD
        .flatMap(_.split(" "))
        .flatMap(_.split("\t"))
        .filter(_.nonEmpty)
        .map(w => (w, 1))
        .countByKey()
        .values
        .sum
    sc.stop()
```

```

if (localWordCount == dfsWordCount) {
    println(s"Success! Local Word Count ($localWordCount) " +
        s"and DFS Word Count ($dfsWordCount) agree.")
} else {
    println(s"Failure! Local Word Count ($localWordCount) " + s"and DFS
Word Count ($dfsWordCount) disagree.")
}
}

private def printUsage(): Unit = {
    val usage: String = "DFS Read-Write Test\n" +
        "\n" + "Usage: localFile dfsDir\n" + "\n" +
        "localFile - (string) local file to use in test\n" +
        "dfsDir - (string) DFS directory for read/write tests\n"
    println(usage)
}

private def readFile(filename: String): List[String] = {
    val liner: Iterator[String] = fromFile(filename).getLines()
    val lineList: List[String] = liner.toList
    lineList
}

def runLocalWordCount(fileContents: List[String]): Int = {
    fileContents.flatMap(_.split(" "))
        .flatMap(_.split("\t"))
        .filter(_.nonEmpty)
        .groupBy(w => w)
        .mapValues(_.size)
}

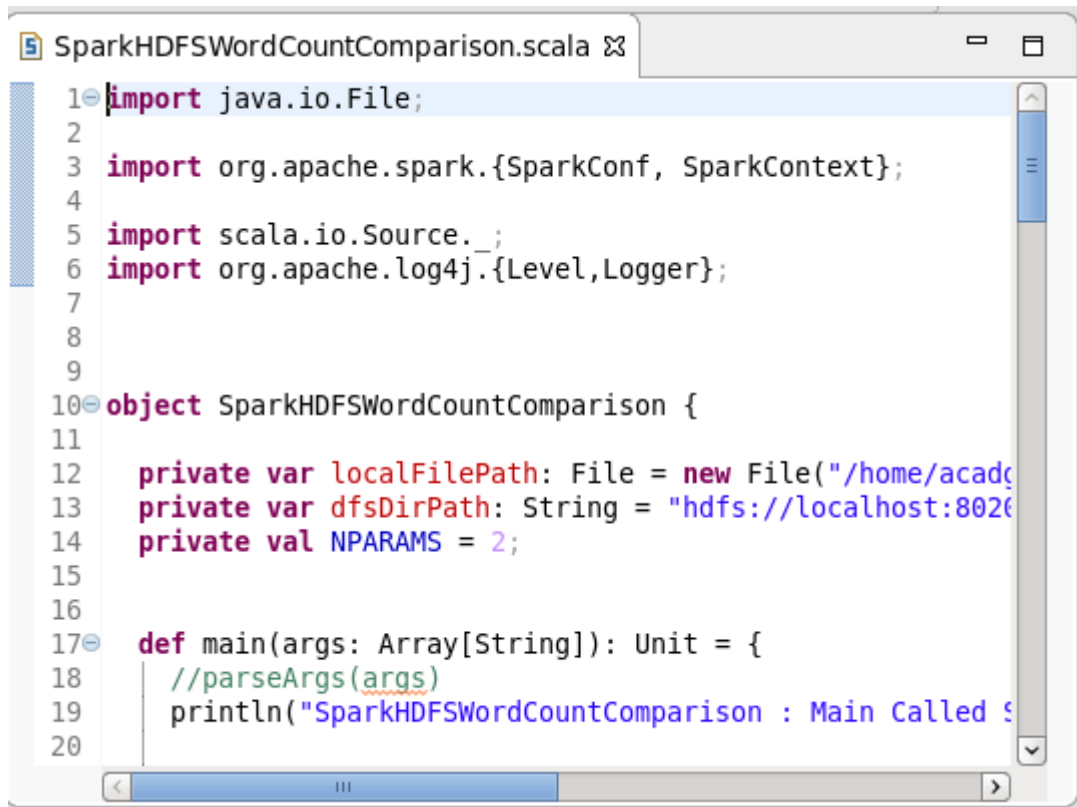
```

.values

.sum

}

}



```
1 import java.io.File;
2
3 import org.apache.spark.{SparkConf, SparkContext};
4
5 import scala.io.Source._;
6 import org.apache.log4j.{Level, Logger};
7
8
9
10 object SparkHDFSWordCountComparison {
11
12     private var localFilePath: File = new File("/home/acadg
13     private var dfsDirPath: String = "hdfs://localhost:8020
14     private val NPARAMS = 2;
15
16
17     def main(args: Array[String]): Unit = {
18         //parseArgs(args)
19         println("SparkHDFSWordCountComparison : Main Called")
20     }
```

```
SparkHDFSWordCountComparison.scala 21
22 println("Performing local word count");
23 val fileContents = readFile(localFilePath.toString());
24 println("Performing local word count - File Content");
25 val localWordCount = runLocalWordCount(fileContents);
26
27 println("SparkHDFSWordCountComparison : Main Called");
28
29 println("Performing local word count Completed !!");
30
31 println("Creating Spark Context");
32
33 val conf = new SparkConf().setMaster("local[2]").setAll();
34 val sc = new SparkContext(conf);
35 val rootLogger = Logger.getLogger();
36 rootLogger.setLevel(Level.ERROR);
37
38
39 println("Spark Context Created");
40
```

```
SparkHDFSWordCountComparison.scala 83
41 println("Writing local file to DFS");
42 val dfsFilename = dfsDirPath + "/dfs_read_write_test"
43 val fileRDD = sc.parallelize(fileContents);
44 fileRDD.saveAsTextFile(dfsFilename);
45 println("Writing local file to DFS Completed");
46
47 println("Reading file from DFS and running Word Count");
48 val readFileRDD = sc.textFile(dfsFilename);
49
50 val dfsWordCount = readFileRDD
51   .flatMap(_.split(" "))
52   .flatMap(_.split("\\t"))
53   .filter(_.nonEmpty)
54   .map(w => (w, 1))
55   .countByKey()
56   .values
57   .sum;
58
59 sc.stop();
60
```

```
SparkHDFSWordCountComparison.scala 83
60
61 if (localWordCount == dfsWordCount) {
62   println(s"Success! Local Word Count ($localWordCount)
63     s"and DFS Word Count ($dfsWordCount) agree.")
64 } else {
65   println(s"Failure! Local Word Count ($localWordCount)
66     s"and DFS Word Count ($dfsWordCount) disagree.")
67 }
68
69
70
71 }
72
73 /**private def parseArgs(args: Array[String]): Unit =
74   if (args.length != NPARAMS) {
75     printUsage()
76     System.exit(1)
77   }
78   */
79
80
```

Smart Insert 1 · 1

```
SparkHDFSWordCountComparison.scala 83
93     val lineList: List[String] = lineIter.toList
94     lineList
95 }
96
97 def runLocalWordCount(fileContents: List[String]): Int
98   fileContents.flatMap(_.split(" "))
99   .flatMap(_.split("\t"))
100   .filter(_.nonEmpty)
101   .groupBy(w => w)
102   .mapValues(_.size)
103   .values
104   .sum
105 }
106
107
108 }
109
110
```

Step 2: Create a file in the local directory

cat>test.txt

Hey Acadgild

```
acadgild@localhost:~
[acadgild@localhost ~]$ cat>test.txt
Hey Acadgild
^C
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

Step 3: Display Results:

```
Problems Tasks Console
<terminated> SparkHDFSWordCountComparison$ [Scala Application] /usr/java/jdk1.8.0
SparkHDFSWordCountComparison : Main Called Successfully
Performing local word count
Performing local word count - File Content ->>List(Hey acadgild)
SparkHDFSWordCountComparison : Main Called Successfully -> Local Word C
Performing local word count Completed !!
Creating Spark Context
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.pr
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/spark/spark-2.1
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.1
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an exp
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/05/15 20:17:30 INFO SparkContext: Running Spark version 2.2.1
18/05/15 20:17:33 WARN NativeCodeLoader: Unable to load native-hadoop l
18/05/15 20:17:34 WARN Utils: Your hostname, localhost.localdomain reso
18/05/15 20:17:34 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to
18/05/15 20:17:35 INFO SparkContext: Submitted application: SparkHDFSWo
```

```
Problems Tasks Console
<terminated> SparkHDFSWordCountComparison$ [Scala Application] /usr/java/jdk1.8.0
18/05/15 20:17:39 INFO SparkEnv: Registering OutputCommitCoordinator
18/05/15 20:17:41 INFO Utils: Successfully started service 'SparkUI' on
18/05/15 20:17:42 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started a
18/05/15 20:17:43 INFO Executor: Starting executor ID driver on host lo
18/05/15 20:17:43 INFO Utils: Successfully started service 'org.apache.
18/05/15 20:17:43 INFO NettyBlockTransferService: Server created on 192
18/05/15 20:17:43 INFO BlockManager: Using org.apache.spark.storage.Ran
18/05/15 20:17:43 INFO BlockManagerMaster: Registering BlockManager Blo
18/05/15 20:17:43 INFO BlockManagerMasterEndpoint: Registering block ma
18/05/15 20:17:43 INFO BlockManagerMaster: Registered BlockManager Blo
18/05/15 20:17:43 INFO BlockManager: Initialized BlockManager: BlockMan
Spark Context Created
Writing local file to DFS
Writing local file to DFS Completed
Reading file from DFS and running Word Count
Success! Local Word Count (2) and DFS Word Count (2) agree.
```