Case Study - V

Spark Streaming 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.

Answer: Now follow the below steps to continue and complete the case study.

1. Create a folder in local file system "/home/acadgild/StreamingInput" and keep it empty as of now.
2. Now run the below code in Eclipse after removing all the compilation errors. Here the Streaming will be done in every 15 seconds of time.

SparkFileStreamingWordCount.scala | NetworkWordCount.scala

```scala
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.log4j.{Level,Logger}


object SparkFileStreamingWordCount {

  def main(args: Array[String]): Unit = {
    println("hey Spark Streaming")

    val conf = new SparkConf().setMaster("local[2]").setAppName("SparkSteamingExample")
    val sc = new SparkContext(conf)
val rootLogger =Logger.getRootLogger()
   rootLogger.setLevel(Level.ERROR)
    val ssc = new StreamingContext(sc, Seconds(15))
    //val lines = ssc.textFileStream(args(0))
    val lines = ssc.textFileStream("/home/acadgild/StreamingInput/")
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
    wordCounts.print()
    ssc.start()
    ssc.awaitTermination()


  }

}
```

3. In the above code update the input argument as the location of local file system directory "/home/acadgild/StreamingInput".(refer above screenshot)
4. Now in terminal use nano command to create new files in the above mentioned location and input some words into it.

```
[acadgild@localhost StreamingInput]$ nano streamtest1.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost StreamingInput]$ nano streamtest2.txt
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost StreamingInput]$ nano streamtest3.txt
[acadgild@localhost StreamingInput]$ cat streamtest3.txt
This is the third file placed in the local machine for the Spark streaming example using a file.
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost StreamingInput]$ nano streamtest3.txt
[acadgild@localhost StreamingInput]$ nano streamtest3.txt
[acadgild@localhost StreamingInput]$
```

5. Here in the output console of eclipse we could see the word count of the files which is updated every 15 seconds is automated.
   Output of file 1

```
Problems  Tasks  Console
<terminated> SparkFileStreamingWordCount$ [Scala Appli

-------------------------------------------
Time: 1529145660000 ms
-------------------------------------------
(is,1)
(This,1)
(first,1)
(Streaming,1)
(file,2)
(Spark,1)
(a,1)
(on,1)
(local,1)
(in,1)
...

-------------------------------------------
Time: 1529145675000 ms
-------------------------------------------

-------------------------------------------
Time: 1529145690000 ms
-------------------------------------------

-------------------------------------------
Time: 1529145705000 ms
```

Output of file 2

```
Problems  Tasks  Console
<terminated> SparkFileStreamingWordCount$ [Scala

-------------------------------------------
Time: 1529145705000 ms
-------------------------------------------

-------------------------------------------
Time: 1529145720000 ms
-------------------------------------------
(is,1)
(second,1)
(This,1)
(exampple,1)
(Streaming,1)
(file,1)
(Spark,1)
(local,1)
(in,1)
(for,1)
...

-------------------------------------------
Time: 1529145735000 ms
-------------------------------------------
```

Output of file 3

```
Problems  Tasks  Console
<terminated> SparkFileStreamingWordCount$ [Scala Appl
-------------------------------------------

-------------------------------------------
Time: 1529145765000 ms
-------------------------------------------
(example,1)
(is,1)
(This,1)
(using,1)
(streaming,1)
(file,1)
(file.,1)
(Spark,1)
(a,1)
(local,1)
...

-------------------------------------------
Time: 1529145780000 ms
-------------------------------------------

-------------------------------------------
Time: 1529145795000 ms
```

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

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

Answer: Follow the below steps to complete the second part.

1. In the Scala IDE write/ copy the below code and clear all the compilation errors. All the steps mentioned in the requirement is taken up in the scala code attached here.

```
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/Documents/s26/usecase/inputs/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("SparkHDFSWordCountCompari
sonApp")
    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 parseArgs(args: Array[String]): Unit = {
    if (args.length != NPARAMS) {
      printUsage()
      System.exit(1)
    }
  }***/

  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 lineIter: Iterator[String] = fromFile(filename).getLines()
    val lineList: List[String] = lineIter.toList
    lineList
  }

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

2. The above code will have two inputs – first: provide the local file location and next is provide the HDFS file location in the above code attached.
3. Now run the code as Scala Application and get the below mentioned output.

```
Problems  Tasks  Console
<terminated> SparkHDFSWordCountComparison$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Jun 16, 2018, 4:47:28 PM)
SparkHDFSWordCountComparison : Main Called Successfully
Performing local word count
Performing local word count - File Content ->>List(This is the first file for Spark Streaming on a file placed in l
SparkHDFSWordCountComparison : Main Called Successfully -> Local Word Count is ->>15
Performing local word count Completed !!
Creating Spark Context
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/spark/spark-2.2.1-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/i
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/06/16 16:47:30 INFO SparkContext: Running Spark version 2.2.1
18/06/16 16:47:31 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-ja
18/06/16 16:47:31 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; usinc
```

.

.

.

```
18/06/16 16:47:33 INFO BlockManagerMaster: Registering BlockManager BlockManager
18/06/16 16:47:33 INFO BlockManagerMasterEndpoint: Registering block manager 192
18/06/16 16:47:33 INFO BlockManagerMaster: Registered BlockManager BlockManagerI
18/06/16 16:47:33 INFO BlockManager: Initialized BlockManager: BlockManagerId(dr
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 (15) and DFS Word Count (15) agree.
```