Session 20 : SPARK Streaming

Assignment 1

**Problem Statement**

Read a stream of Strings, fetch the words which can be converted

to numbers. Filter out the rows, where the sum of numbers in that

line is odd.

Provide the sum of all the remaining numbers in that batch.

**Solution:-**

Project - 

Scala classes -  

Project JAR file - 

Main logic is in EvenLines.scala

Code:-

**EvenLines.scala**

package org.scala

import org.apache.spark.SparkConf

import org.apache.spark.storage.StorageLevel

import org.apache.spark.streaming.{Seconds, StreamingContext}

/\*\*

\* Counts words in UTF8 encoded, '\n' delimited text received from the network every second.

\*Usage: EvenLines <hostname> <port>

\* <hostname> and <port> describe the TCP server that Spark Streaming would connect to receive data.

\*

\* To run this on your local machine, you need to first run a Netcat server

\* `$ nc -lk 9999`

\* and then run the example

\* `$ bin/run-example org.apache.spark.examples.streaming.NetworkWordCount localhost 9999`

\*/

object EvenLines {

def main(args: Array[String]) {

if (args.length < 2) {

System.err.println("Usage: EvenLines <hostname> <port>")

System.exit(1)

}

StreamingExamples.setStreamingLogLevels()

// Create the context with a 10 second batch size

val sparkConf = new SparkConf().setAppName("EvenLines")

val ssc = new StreamingContext(sparkConf, Seconds(10))

var strList = "";

// Create a socket stream on target ip:port and count the

// words in input stream of \n delimited text (eg. generated by 'nc')

// Note that no duplication in storage level only for running locally.

// Replication necessary in distributed scenario for fault tolerance.

val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY\_AND\_DISK\_SER);

val overAllList = List("");

val tempList = List("");

val linesFiltered = lines.filter { x => getLineSum(x)%2==0 };

val linesSum = linesFiltered.map { x => getLineSum(x) };

println("Lines with even sum");

linesFiltered.print();

println("");

print("Sum of numbers in even lines : ");

linesSum.reduce( (c1, c2) => c1 + c2).print();

ssc.start()

ssc.awaitTermination()

}

def getLineSum(ln : String): Double={

val lineWords = ln.split(" ");

var num: Double = 0;

for(x <- lineWords)

{

try {

val f = x.toDouble;

num = num + f;

} catch {

case ex: Exception =>{ }

}

}

return num; }}

**StreamingExamples.scala**

package org.scala

import org.apache.log4j.{Level, Logger}

import org.apache.spark.internal.Logging

/\*\* Utility functions for Spark Streaming examples. \*/

object StreamingExamples extends Logging {

/\*\* Set reasonable logging levels for streaming if the user has not configured log4j. \*/

def setStreamingLogLevels() {

val log4jInitialized = Logger.getRootLogger.getAllAppenders.hasMoreElements

if (!log4jInitialized) {

// We first log something to initialize Spark's default logging, then we override the

// logging level.

logInfo("Setting log level to [WARN] for streaming example." +

" To override add a custom log4j.properties to the classpath.")

Logger.getRootLogger.setLevel(Level.WARN)

}

}

}

**-----------------------------------------------Running the JAR---------------------------------------------------**

[acadgild@localhost Assignment20\_1]$ spark-submit --master local[2] -class org.scala.EvenLines Assignment20\_1.jar localhost 9999



Running netcat at port 9999 for input

[acadgild@localhost Assignment20\_1]$ nc -lk 9999

Sample input –

test 2 3 5

test 7

test 9

100

Expected output will be

Test 2 3 5

10.0

100

100.0

Got the same output in spark job. PFB screenshot



Odd sum lines test 7 and test 9 are ignored and even sum lines and their sum is printed.