Question

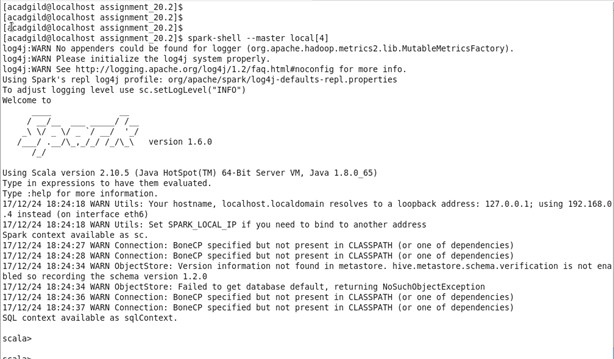
Read a stream of Strings, fetch the words which can be converted to numbers. Filter out the rows, where the sum of numbers in that line is odd.

Provide the sum of all the remaining numbers in that batch. Solution:-

Step1: Start Spark Shell with 4 threads

Use the command below to start spark-shell with 4 threads spark-shell --master local[4]

Screenshot is as below:



- Step2: Declare all the packages

Declare spark and streaming packages as below:

import org.apache.spark.\_

import org.apache.spark.streaming.\_

import org.apache.spark.streaming.StreamingContext.\_

Step3: Declare a accumulator

Declare accumulator totalEvenLinesWordNumber which will keep track of sum of number of word numbers in lines so far

val totalEvenLinesWordNumber = sc.accumulator(0)

Step4: Define a wordNumberMap map for converting word to number

Define a map for converting word to number. If word is not there in map then 0 will be returned. Broadcast the map. Code is as below

val wordNumberMap = Map("Hi" -> 1, "my" -> 2, "name" -> 3, "is" -> 4, "Hello" -

>5, "Monimoy" -> 6, "John" -> 7, "Bob"->8, "Vibhu" ->9)

val wordNumberMapBroadcast = sc.broadcast(wordNumberMap)

Step4: Define a function to return sum of word converted to number in a line

Define a function lineWordNumberTotal which will split a line based on blank space to get all the words in a next. Next in the lookup wordNumberMapBroadcast , based on word, corresponding number is retrieved and sum all these numbers together.

Code is as below:

def lineWordNumberTotal(line:String):Int = { var sum:Int = 0

var words = line.split(" ")

for (word <- words) sum += wordNumberMapBroadcast.value.get(word).getOrElse(0)

sum

}

Screenshot of steps 2 to 4 is as below:



Step5: Start Text Streaming

- Start text streaming on localhost with port number 9999 and interval 15 seconds and return the stream. Code is as below:

val ssc = new StreamingContext(sc, Seconds(15))

val stream = ssc.socketTextStream("localhost", 9999)

Step6: Process each RDD in stream

Process each RDD in stream. First convert the RDD to string. If it is not blank calculate word number for each word and sum them using function

lineWordNumberTotal and put to variable numTotal. If numTotal is odd, print the corresponding line. Also, add numTotal to accumulator accu totalEvenLinesWordNumber and print the sum

stream.foreachRDD(line => {

val lineStr = line.collect().toList.mkString("") if (lineStr != "") {

var numTotal = lineWordNumberTotal(lineStr) if (numTotal % 2 == 1) println(lineStr)

else {

totalEvenLinesWordNumber += numTotal

println("Sum of lines with even word number so far =" + totalEvenLinesWordNumber.value.toInt)

}

}

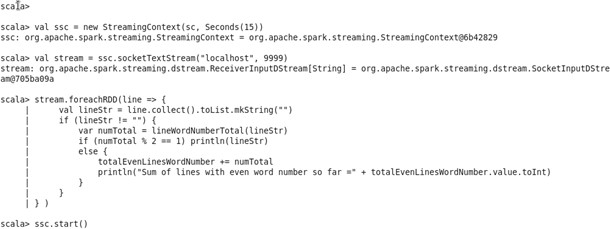
} )

Step7: Start the streams

Start the streams and wait till its termination. Code is as below: ssc.start()

ssc.awaitTermination()

Screenshot for steps 5 to 7 is as below:

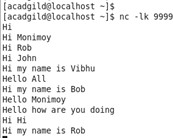


Step8: Start netcat from a terminal

From a terminal start netcat on port 9999 using nc ommand below and start typing lines

nc -lk 9999

The screenshot is as below:



Step9: Display the output

The lines with odd numbered word number sum will be displayed. For lines with even numbered word number, the summation done so far will be displayed. The screenshot is as below:

For example

“Hi Monimoy” has total word number value 7 which is odd, so the line will be displayed

“Hi John” has total word number value 8 which is even number so summation will be displayed

