Read two streams

1. List of strings input by user

2. Real-time set of offensive words

Find the word count of the offensive words inputted by the user as per the

real-time set of offensive words.

Cd workspace-eclipse

Cd wordcountstreaming

Sbt package

Sbt eclipse

***import org.apache.spark.\_***

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

***object NetworkWordCount {***

***def main(args:Array[String]) {***

***val SparkConf = new SparkConf().setAppName("NetworkWordCount").setMaster("local[2]")***

***// Create a local StreamingContext with batch interval of 10 second***

***val ssc = new StreamingContext(SparkConf, Seconds(10))***

***/\* Create a DStream that will connect to hostname and port, like localhost 9999. As stated earlier, DStream will get created from StreamContext, which in return is created from SparkContext. \*/***

***val lines = ssc.socketTextStream("localhost",9999)***

***// Using this DStream (lines) we will perform transformation or output operation.***

***val words = lines.flatMap(\_.split(" "))***

***val wordCounts = words.map(x => (x, 1)).reduceByKey(\_ + \_)***

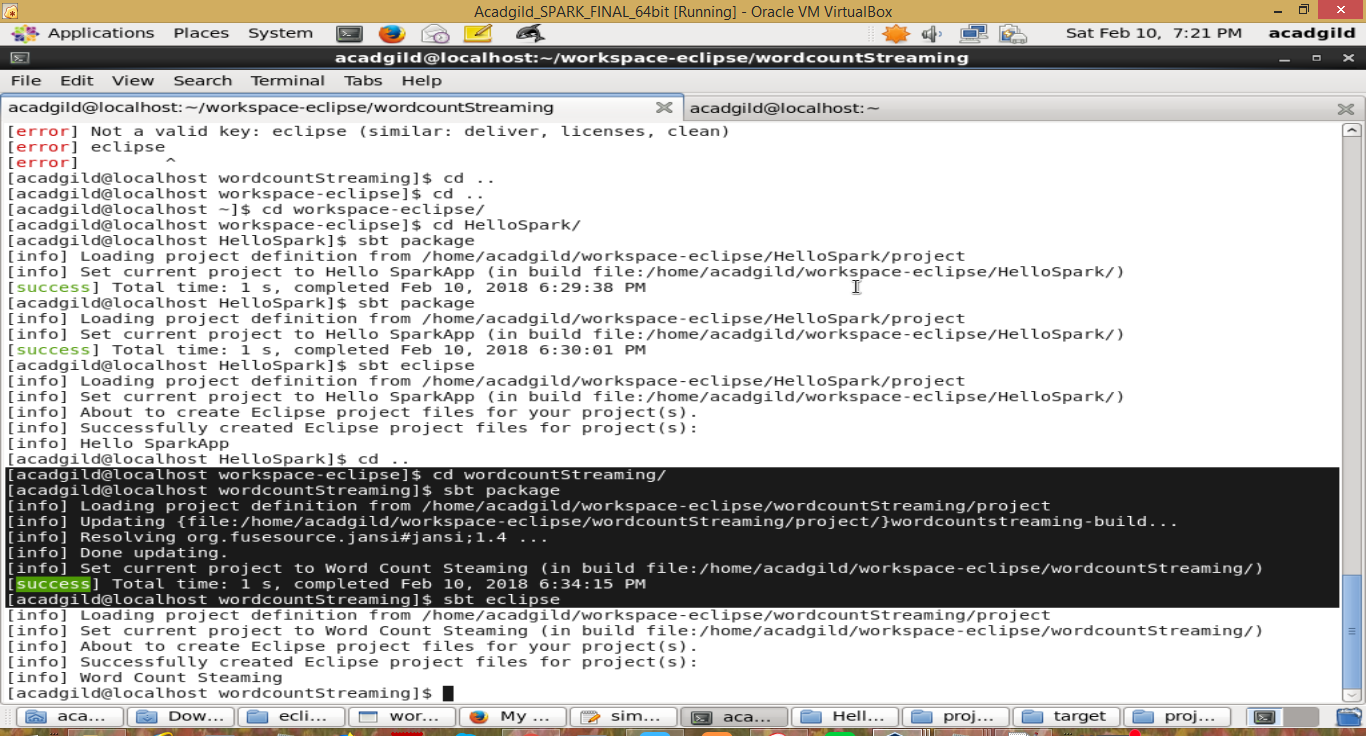
***wordCounts.print()***

***ssc.start() // Start the computation***

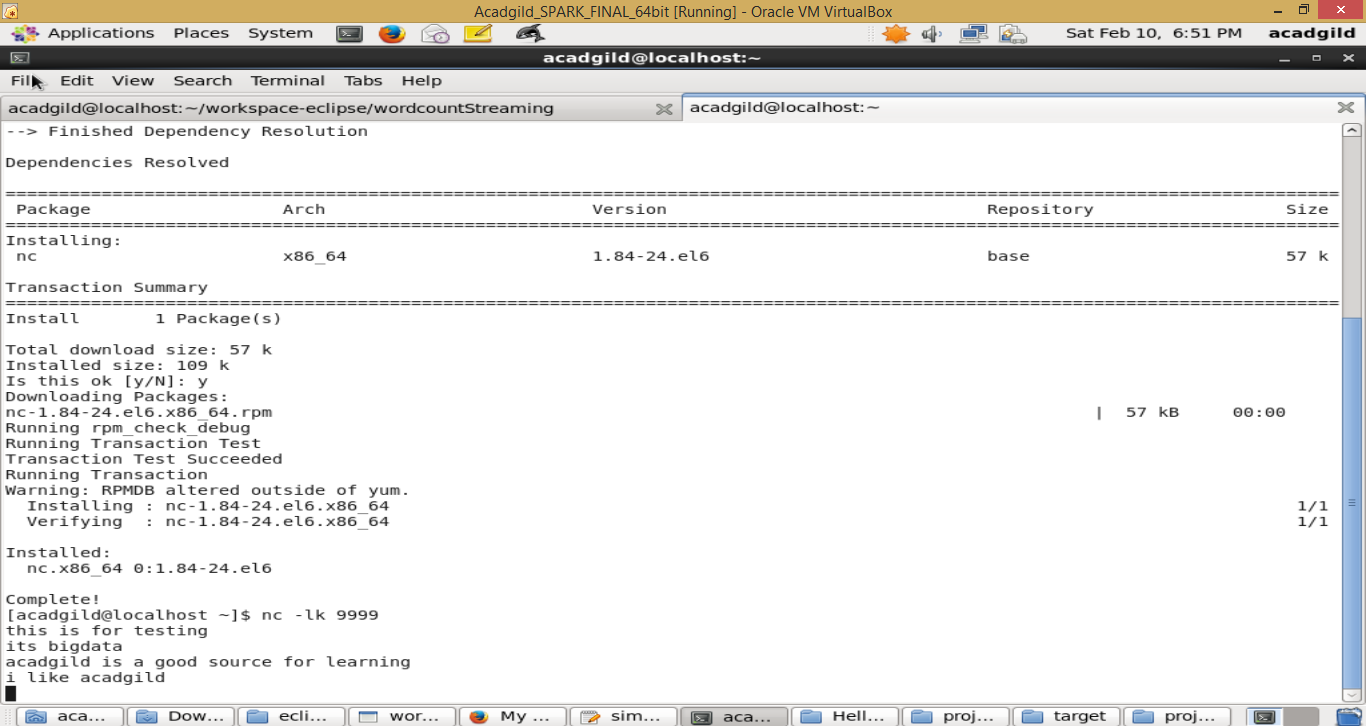
***ssc.awaitTermination() // Wait for the computation to terminate***

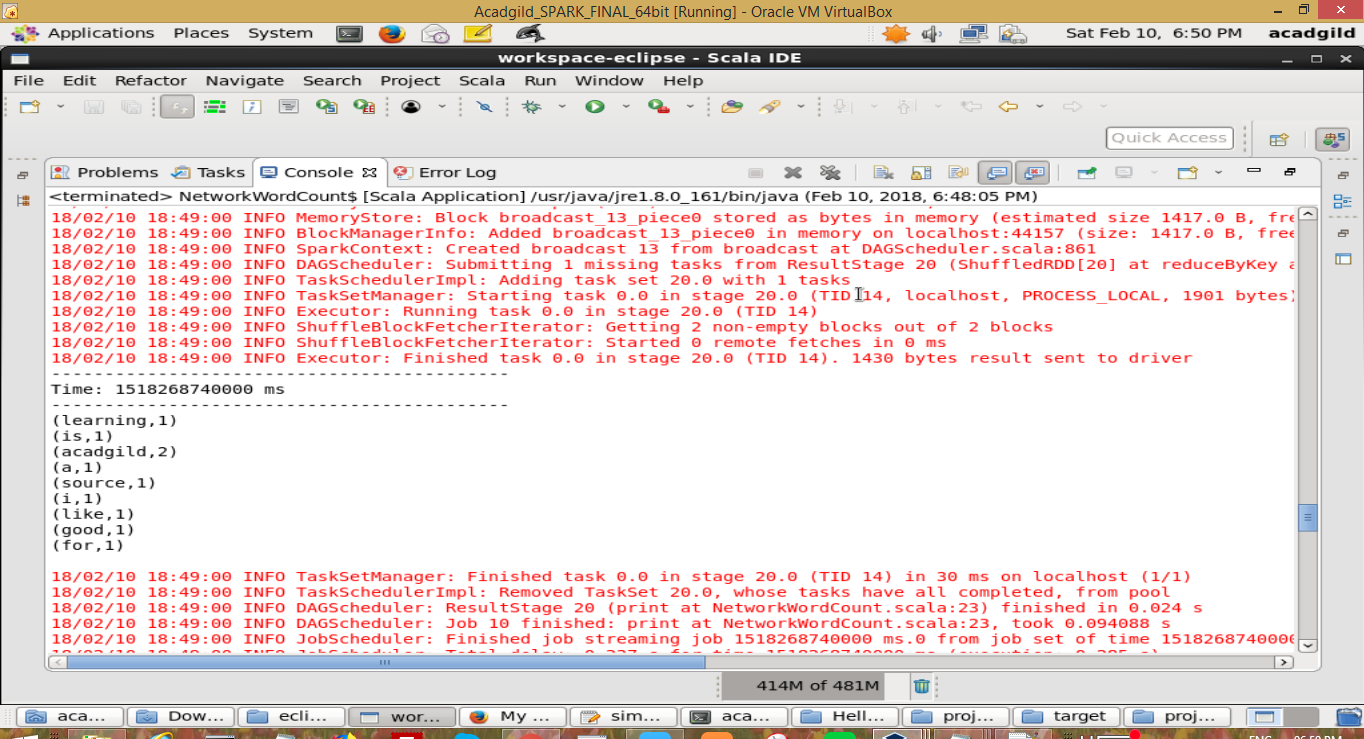
***}***

***}***



Stateless





Statefull

***import org.apache.spark.\_***

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

***object NetworkWordCount\_stateful {***

***def main(args:Array[String]) {***

***val SparkConf = new SparkConf().setAppName("NetworkWordCount").setMaster("local[2]")***

***// Create a local StreamingContext with batch interval of 10 second***

***val ssc = new StreamingContext(SparkConf, Seconds(10))***

***/\* Create a DStream that will connect to hostname and port, like localhost 9999. As stated earlier, DStream will get created from StreamContext, which in return is created from SparkContext. \*/***

***val lines = ssc.socketTextStream("localhost",9999)***

***// Using this DStream (lines) we will perform transformation or output operation.***

***val words = lines.flatMap(\_.split(" "))***

***//val wordCounts = words.map(x => (x, 1)).reduceByKey(\_ + \_)***

***val wordCounts = words.map(x => (x, 1)).reduceByKeyAndWindow(\_+\_, Seconds(30))***

***wordCounts.print()***

***ssc.start() // Start the computation***

***ssc.awaitTermination() // Wait for the computation to terminate***

***}***

***}***

