## HU Extension Assignment 03 E63 Big Data Analytics

### Handed out: 09/15/2017 Due by 11:59 AM EST on Saturday, 09/23/2017

### Use either VMWare Workstation or VMWare Fusion

[https://e5.onthehub.com/WebStore/Welcome.aspx?ws=4185a0dc-d0d1-e511-9416-b8ca3a5db7a1&vsro=8](https://e5.onthehub.com/WebStore/Welcome.aspx?ws=4185a0dc-d0d1-e511-9416-b8ca3a5db7a1&vsro=8" \t "_blank)

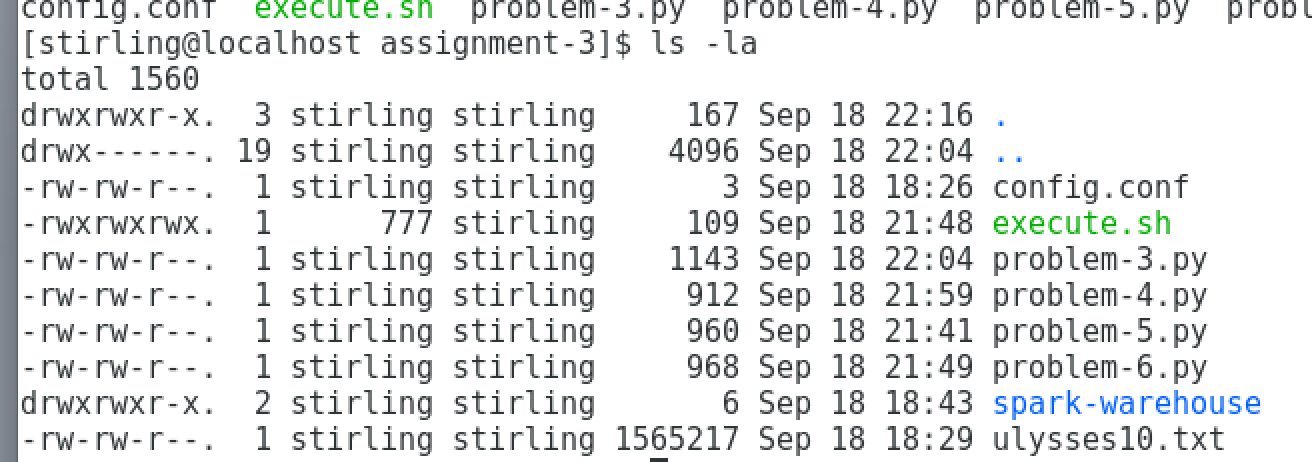
If you feel comfortable with a virtualization technology different from VMWare, please be free to use it. Rather than creating a VM with CentOS, you can create the VM with a different flavor of Linux you more familiar with or consider better.

**Problem 1**. Create your own Virtual Machine with a Linux operating system. The lecture notes speak about CentOS. You are welcome to work with another Linux OS. When creating the VM, create an administrative user. Call that user whatever you feel like. Please record the password of the new user. Once the VM is created transfer the attached text file Ulysses10.txt to the home of new user. You can do it using scp (secure copy command) or email. Examine the version of Java, Python and Scala on your VM. If any of those versions is below requirements for Spark 2.2 install proper version. Set JAVA\_HOME environmental variable. Set your PATH environmental variable properly, so that you can invoke: java, sbt and python commands from any directory on your system. [20%]

/opt2/spark-2.2.0-bin-hadoop2.7/sbin/start-master.sh

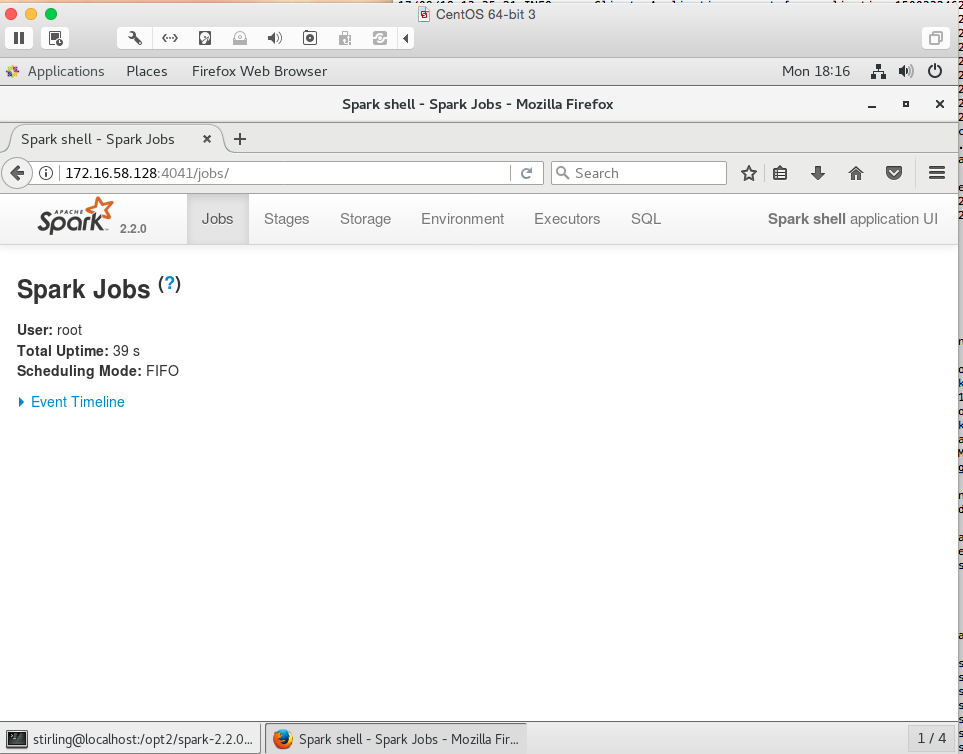
/opt2/spark-2.2.0-bin-hadoop2.7/bin/spark-submit

|  |
| --- |
| **# Set everything to be logged to the console**  log4j.rootCategory=ERROR, console  log4j.appender.console=org.apache.log4j.ConsoleAppender  log4j.appender.console.target=System.err  log4j.appender.console.layout=org.apache.log4j.PatternLayout  log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n |



**Problem 2**. Install Spark 2.2 on your VM. Make sure that pyspark is also installed. Demonstrate that you can successfully open spark-shell and that you can eliminate most of WARNing messages. [15%]

|  |
| --- |
| [stirling@localhost sbin]$ sudo spark-shell  Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties  Setting default log level to "WARN".  To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  17/09/18 18:09:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  17/09/18 18:09:57 WARN Utils: Your hostname, localhost.localdomain resolves to a loopback address: 127.0.0.1; using 172.16.58.128 instead (on interface ens33)  17/09/18 18:09:57 WARN Utils: Set SPARK\_LOCAL\_IP if you need to bind to another address  17/09/18 18:09:57 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.  17/09/18 18:10:01 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0  17/09/18 18:10:01 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException  17/09/18 18:10:02 WARN ObjectStore: Failed to get database global\_temp, returning NoSuchObjectException  Spark context Web UI available at http://172.16.58.128:4041  Spark context available as 'sc' (master = local[\*], app id = local-1505772597695).  Spark session available as 'spark'.  Welcome to  \_\_\_\_ \_\_  / \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_  \_\ \/ \_ \/ \_ `/ \_\_/ '\_/  /\_\_\_/ .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.2.0  /\_/    Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0\_144)  Type in expressions to have them evaluated.  Type :help for more information.  scala> |



**Problem 3**. Find the number of lines in the text file ulysses10.txt that contain word “afternoon” or “night” or “morning”. In this problem use RDD API. Do this in two ways, first create a lambda function which will test whether a line contains any one of those 3 words. Second, create a named function in the language of choice that returns TRUE if a line passed to it contains any one of those three words. Demonstrate that the count is the same. Use pyspark and Spark Python API. If convenient you are welcome to implement this problem in any other language: Scala, Java or R. [15%]

**# create a lambda function which will test whether a line contains any one of those 3 words.**

|  |
| --- |
| lines = sc.textFile("file:////home/stirling/assignment-3/ulysses10.txt")  wordLines = lines.filter(lambda line: "afternoon" in line or "night" in line or "morning" in line)  print(wordLines.count()) |

**Answer**

|  |
| --- |
| **> 418** |

**# Second, create a named function in the language of choice that returns TRUE if a line passed to it contains any one of those three words. Demonstrate that the count is the same.**

|  |
| --- |
| def hasWords(line):  return "afternoon" in line or "night" in line or "morning" in line  lines = sc.textFile("file:////home/stirling/assignment-3/ulysses10.txt")  wordLines = lines.filter(hasWords)  print(wordLines.count()) |

**ANSWER**

|  |
| --- |
| **> 418** |

**Problem 4.** Implement the above task, finding the number of lines with one of those three words in file ulysses10.txt using Dataset/DataFrame API. Again, use the language of your choice. [20%]

|  |
| --- |
| dset = spark.read.text("file:////home/stirling/assignment-3/ulysses10.txt")  print(dset.count())  wordLines = dset.filter(dset.value.contains('afternoon') | dset.value.contains('night') | dset.value.contains('morning'))  print(wordLines.count()) |

**ANSWER**

|  |
| --- |
| **> 418** |

**Problem 5**. Create a standalone Python script that will count all words in file ulysses10.txt. You are expected to produce a single number. Do it using RDD API. If convenient, you are welcome to implement this problem in other languages: Scala, Java or R. [%15]

|  |
| --- |
| from pyspark import SparkConf, SparkContext, SQLContext  from pyspark.sql import SQLContext, SparkSession, Row  from pyspark.sql.types import \*  from pyspark.sql.functions import \*  conf = (  SparkConf()  .setAppName("assignment-3")  .set("spark.executor.instances", 1)  .set("spark.executor.cores", 1)  .set("spark.shuffle.compress", "true")  .set("spark.io.compression.codec", "snappy")  .set("spark.executor.memory", "4g")  )  sc = SparkContext().getOrCreate(conf = conf)  sc.setLogLevel("ERROR")  sqlContext = SQLContext(sc)  spark = SparkSession.builder.appName("spark play").getOrCreate()  word\_file = sc.textFile("file:////home/stirling/assignment-3/ulysses10.txt")  word\_count = word\_file.flatMap(lambda x: "".join(x).encode("utf-8", "ignore").strip().split()).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).values().sum()  print("Total Word Count: {0}".format(str(word\_count))) |

**Answer**

|  |
| --- |
| **> Total Word Count: 267,832** |

**Problem 6.** Create a standalone Python script that will count all words in file ulysses10.txt. You are expected to produce a single number. Do it using Dataset/DataFrame API. If convenient, you are welcome to implement this problem in other languages: Scala, Java or R. [%15]

|  |
| --- |
| from pyspark import SparkConf, SparkContext, SQLContext  from pyspark.sql import SQLContext, SparkSession, Row  from pyspark.sql.types import \*  from pyspark.sql.functions import \*  conf = (  SparkConf()  .setAppName("assignment-3")  .set("spark.executor.instances", 1)  .set("spark.executor.cores", 1)  .set("spark.shuffle.compress", "true")  .set("spark.io.compression.codec", "snappy")  .set("spark.executor.memory", "4g")  )  sc = SparkContext().getOrCreate(conf = conf)  sc.setLogLevel("ERROR")  sqlContext = SQLContext(sc)  spark = SparkSession.builder.appName("spark play").getOrCreate()  word\_file = spark.read.text("file:////home/stirling/assignment-3/ulysses10.txt")  word\_count = word\_file.rdd.flatMap(lambda x: "".join(x).encode("utf-8", "ignore").strip().split()).map(lambda x: (x,1)).reduceByKey(lambda a,b: a+b).values().sum()  print("Total Word Count: {0}".format(str(word\_count))) |

**Answer**

|  |
| --- |
| **> Total Word Count: 267,832** |

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document with explanations of the purpose of those commands. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE COPY Snippets of your Code but DO NOT EMBED files into your MS Word document. For issues and comments visit the class Discussion Board on Piazza.