CPrE/SE 419: SOFTWARE TOOLS FOR LARGE-SCALE DATA ANALYSIS, SPRING 2019

Purpose

The goal of this lab is to introduce you to Apache Spark, a fast and general engine for big data processing. Spark provides a basic data abstraction called RDD (Resilient Distributed Dataset) which is a collection of elements, stored in a distributed manner across a cluster. Using RDD transformations, Spark is well suited for data processing in pipelines. Spark also provides rich APIs for RDDs so that users can easily operate data in parallel. In addition, users can optionally persist RDDs in memory so that it will speed up computing when reuse the data.

During this lab, you will learn:

- The Spark platform and usage of its API's (in Java)
- Write program with pipelined jobs to analyze network logs

Submission

Create a single zip archive, named by your last name, with the following and hand it in through canvas:

- The output file for each task generated by your program.
- Commented Code for your program. Include all source files needed for compilation.

Examples

We have 2 examples "WordCount" and "StockPrice", that contain some Spark API usages. We have already seen "WordCount" in Hadoop and Pig, the problem is to count the number of occurrences for each distinct word. In Spark, we first use **flatMap()** method to split each line of text into words and each word is as one element in the RDD. Then we use **mapToPair()** method to transform RDD into PairRDD, that converts each element into <key, value> pair where key is the word and value is one. Finally, we use **reduceByKey()** method to sum all the ones to get the number of counts for each word. Note that the function in **reduceByKey()** method is applied in associative manner, like the Combiner in Hadoop. We also provide another example "StockPrice" in the lecture which analyze the stock prices.

For all the API's on RDD and PairRDD, check the links to their javadocs:

https://spark.apache.org/docs/1.6.0/api/java/org/apache/spark/api/java/JavaRDDLike.html

https://spark.apache.org/docs/1.6.0/api/java/org/apache/spark/api/java/JavaPairRDD.html

For more examples in Java:

https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples

Compile and Submit Application

To compile your Java program, you will need to link Spark libraries and also Hadoop libraries to include them in your class path. The **recommended option** to add the libraries to your program is using Maven. Add the following dependency for Spark library into the pom.xml file in your maven project:

```
<dependencies>


      <!-- https://mvnrepository.com/artifact/org.apache.spark/spark-core -->
      <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.11</artifactId>
    <version>1.6.0</version>
      </dependency>


<!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-common -->
      <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-common</artifactId>
      <version>2.6.0</version>
      <scope>provided</scope>
      </dependency>
      <dependency>
      <groupId>jdk.tools</groupId>
      <artifactId>jdk.tools</artifactId>
      <version>1.8.0_131</version>
      <scope>system</scope>
      <systemPath>C:/Program
Files/Java/jdk1.8.0_201/lib/tools.jar</systemPath>
      </dependency>
      </dependencies>
      <properties>
      <maven.compiler.source>1.8</maven.compiler.source>
      <maven.compiler.target>1.8</maven.compiler.target>
      </properties>
```

Experiment 1 (40 points)

In this experiment we will modify the word count example, so that the output is sorted by the number of counts in descending order. We use the Gutenberg corpus, as the testing input to your program. It is on the HDFS at the following location: /cpre419/gutenberg. Include the source code and the snapshot of first 10 lines of your output file in your submission.

Hint: To achieve sorting, you can use the **sortByKey()** method. However, key is the word but we want to sort by the counts. You can use the **mapToPair()** method to swap the key and value. You can use Shakespeare corpus to test your program first because it is a small data.

Experiment 2 (60 points)

We will redo the firewall example in lab 5 on pig, except here we will write pipelined jobs in Spark.

Given two input files:

/cpre419/ip_trace – An IP trace file having information about connections received from different source IP addresses, along with a connection ID and time.

The format of IP trace file is:

<Time> <Connection ID> <Source IP> “>” <Destination IP> <protocol> <protocol dependent data>

/cpre419/raw_block - A file containing the connection IDs that were blocked

The format of block file is:

<Connection ID> <Action Taken>

Your task is to regenerate the log file by combining information from others logs that are available. The lost firewall log should contain details of all blocked connections and should be in the following format.

<Time> <Connection ID> <Source IP> <Destination IP> “Blocked”

A. (30pts) Regenerate the firewall file containing details of all blocked connections. You only need to submit your source code and the snapshot of first 10 lines your generated firewall log.

B. (30pts) Based on the previous program, generate a list of all unique source IP addresses that were blocked and the number of times that they were blocked. This list should be sorted (by the script) by the number of times that each IP was blocked in descending order. Submit your code and the snapshot of first 10 lines of your output file.

You can write part A and part B in the same program.

Installing Spark on Mac IOS:

(If having any question, please contact Ashraf Tahmasbi <tahmasbi@iastate.edu>)

To successfully install spark on your system, you need to download and install the following:

spark-2.4.0-bin-hadoop2.7

scala-2.11.11

sbt-1.2.8

python-3.6.6-macosx10.9

jdk-8u201-macosx-x64.

Note: Home folder for this tutorial is referred to as $HOME or ~.

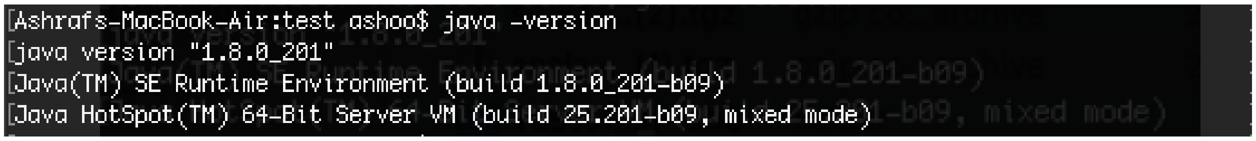
Create a folder names "spark" under your home directory. After downloading all the required files, move and extract them in to this folder.

Install JDK and Python using the downloaded installer. Then, you need to setup shell environment by editing the ~/.bash_profile file. To this end, open the .bash_profile file, which is located at your Home directory using any text editor. If .bash_profile file doesn't exist, create a file named .bash_profile. Open this file and add the following line to it:

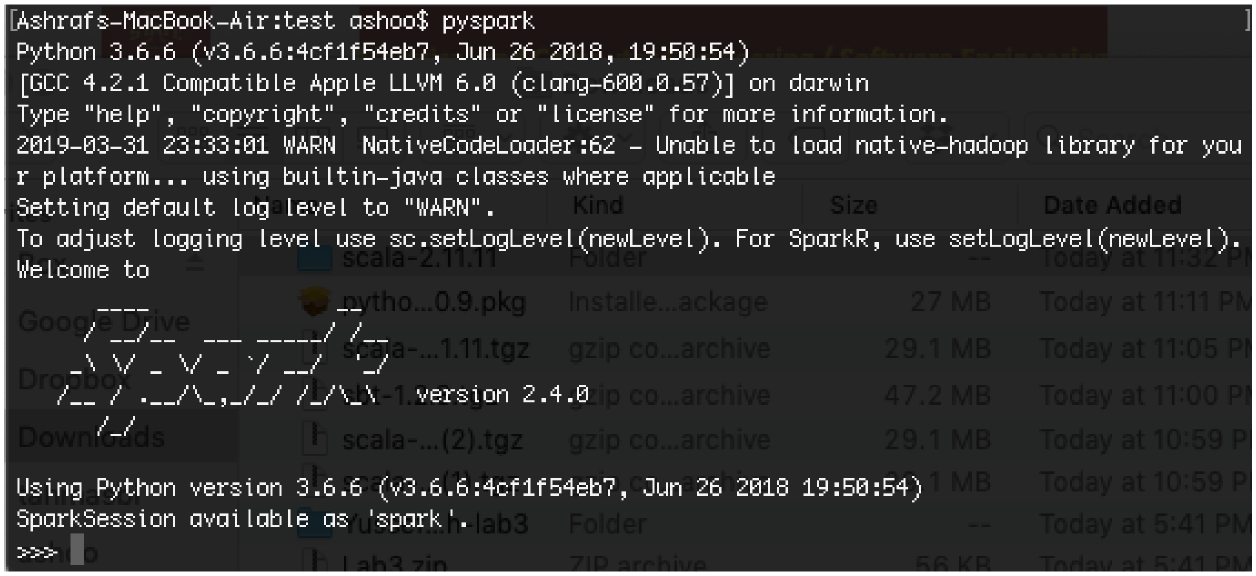
```
export
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/
export SPARK_HOME=~/server/spark-2.4.0-bin-hadoop2.7
export SBT_HOME=~/server/sbt
export SCALA_HOME=~/server/scala-2.11.11
export
PATH=$JAVA_HOME/bin:$SBT_HOME/bin:$SBT_HOME/lib:$SCALA_HOME/bin:$SCALA_HOME/
lib:$PATH
export
PATH=$JAVA_HOME/bin:$SPARK_HOME:$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
export PYSPARK_PYTHON=python3

PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"
export PATH

# Setting PATH for Python 3.6
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"
export PATH
```

After editing .bash_profile file, you have to reload it. To do so type source ~/.bash_profile in your terminal or quit and reopen the terminal program.

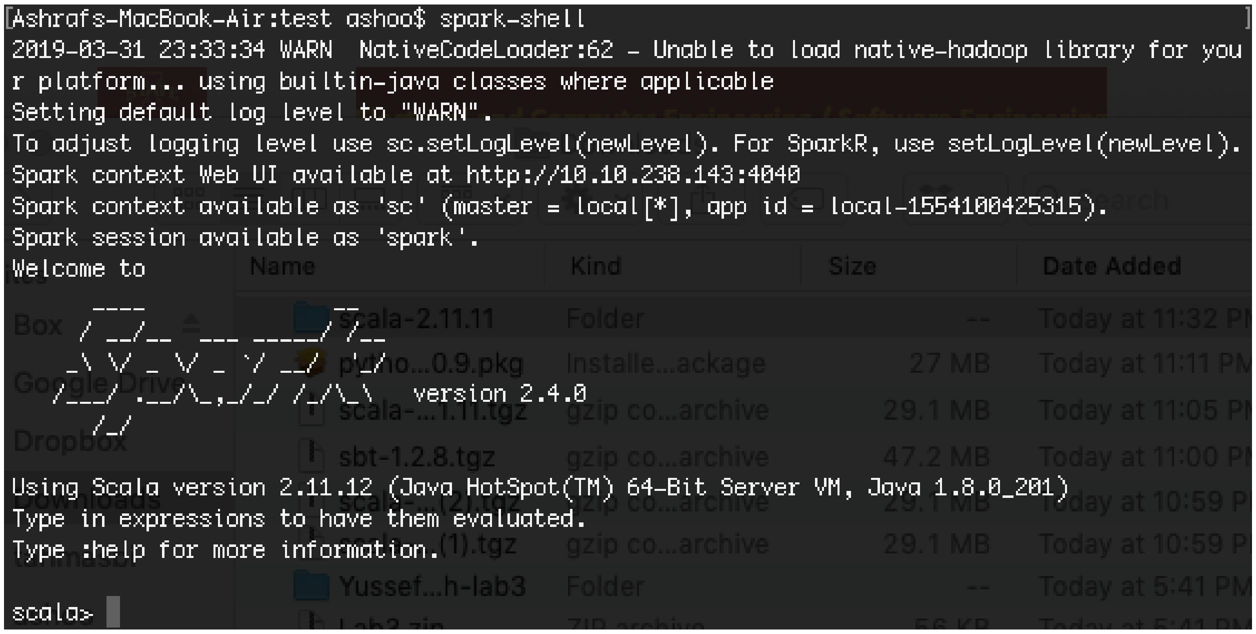
Now, the installation is complete. To test the installation do as follow:



```
[Ashrafs-MacBook-Air:test ashoo$ java -version
[java version "1.8.0_201"
[Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
[Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
```



```
[Ashrafs-MacBook-Air:test ashoo$ pyspark
 Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 26 2018, 19:50:54)
 [GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
 Type "help", "copyright", "credits" or "license" for more information.
 2019-03-31 23:33:01 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for you
 r platform... using builtin-java classes where applicable
 Setting default log level to "WARN".
 To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
 Welcome to
       ____              __
      / __/__  ___ _____/ /__
     _\ \/ _ \/ _ `/ __/  '_/
    /__ / .__/\_,_/_/ /_/\_\   version 2.4.0
       /_/

 Using Python version 3.6.6 (v3.6.6:4cf1f54eb7, Jun 26 2018 19:50:54)
 SparkSession available as 'spark'.
 >>>
```



```
[Ashrafs-MacBook-Air:test ashoo$ spark-shell
2019-03-31 23:33:34 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://10.10.238.143:4040
Spark context available as 'sc' (master = local[*], app id = local-1554100425315).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_201)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Use CTRL-D to quit spark-shell or pyspark shell.

Installing Spark on Windows:

(If having any question, please contact Xu Teng <xuteng@iastate.edu>)

1. Install **Java JDK 1.8.x** for Windows
   Download link: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
   Note: Remember the Path you install the Java (let's call it ***install_JAVA_PATH***)
2. Configure Java
   a. Open ***Control Panel*** -> click ***System and Security*** -> click ***System*** -> click ***Advanced system settings*** -> click ***Environment Variables*** under ***Advanced***
   b. Click ***New…*** under System variables. Add Variable name with ***JAVA_HOME*** and Variable value with ***install_JAVA_PATH***.
   c. Select ***Path*** variable under System variables, click ***Edit…***
   d. Click ***New*** and enter %JAVA_HOME%\bin
3. Configure winutils.exe
   a. Download from link: https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin. You can find winutils.exe under this repository.
   b. Create a new folder under C: drive, named winutils. Then create another new folder, named bin, under C:\winutils. And copy winutils.exe to C:\winutils\bin.
   c. Add new system variable (same as 2.a and 2.b above) whose name is ***HADOOP_HOME*** and value is ***C:\winutils***
4. Configure Spark
   a. Download from link: https://spark.apache.org/downloads.html. Choose Spark release 2.3.3 and Pre-built for Apache Hadoop 2.7 and later. Then click Download Spark, and select one mirror site for download.
   b. Unzip download file and find folder ***spark-2.3.3-bin-hadoop2.7***.
   c. Copy ***spark-2.3.3-bin-hadoop2.7*** to C: drive
   d. Add new system variable (same as 2.a and 2.b above) whose name is ***SPARK_HOME*** and value is ***C:\spark-2.3.3-bin-hadoop2.7***
   e. Select ***Path*** variable under System variables, click ***Edit…***
   f. Click ***New*** and enter %SPARK_HOME%\bin

Eventually, Spark has been installed and configured on our local. Open your terminal and cd into the directory you copied Spark files to (in our case, ***C:\spark-2.3.3-bin-hadoop2.7***). Then type ***spark-shell***.