Indian Institute of Technology Madras
CS6847: Cloud Computing

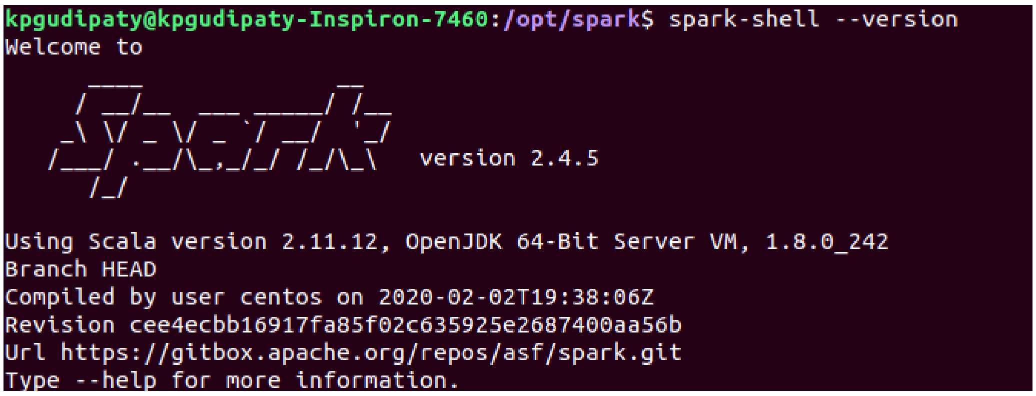
Krishna Praneet - MM16B029

April 19, 2020

Assignment-3 Report

Setup

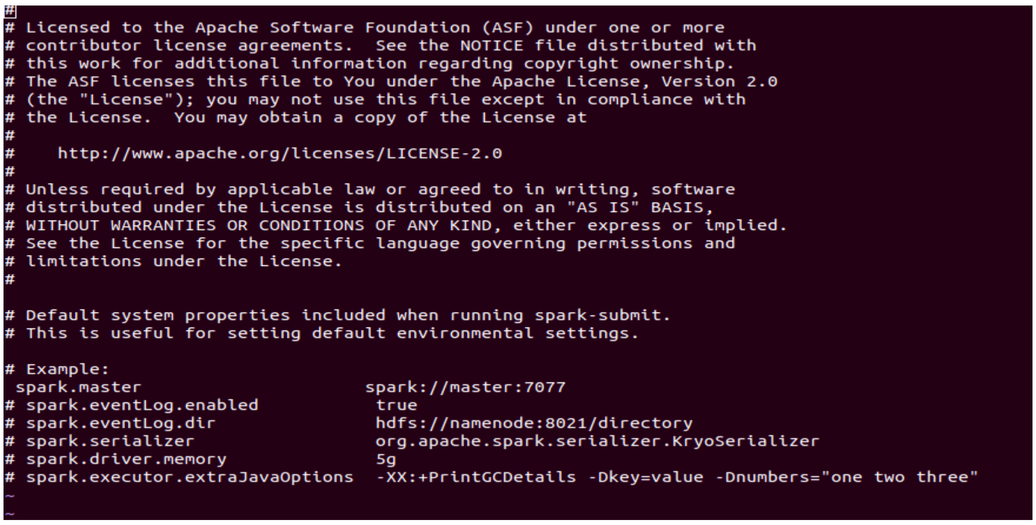
Apache spark has been setup on Ubuntu system. The *tar* file was downloaded from the Apache website and extracted to suitable folder.*Scala* was already installed, along with *Java*. Successful installation can be verified by running `spark-shell --version` which displays the version of the installation as well. Figure 2 shows the configuration file for spark which shows the setup of one master node i.e. the localhost.



```
kpgudipaty@kpgudipaty-Inspiron-7460:/opt/spark$ spark-shell --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.5
      /_/

Using Scala version 2.11.12, OpenJDK 64-Bit Server VM, 1.8.0_242
Branch HEAD
Compiled by user centos on 2020-02-02T19:38:06Z
Revision cee4ecbb16917fa85f02c635925e2687400aa56b
Url https://gitbox.apache.org/repos/asf/spark.git
Type --help for more information.
```

Figure 1: Spark successful installation



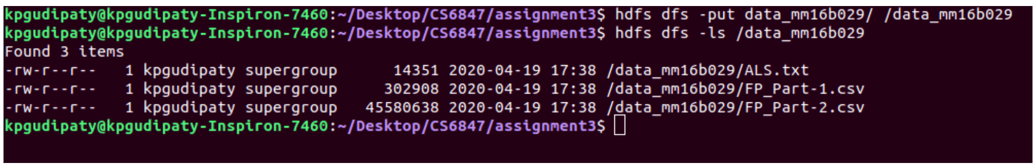
```
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Default system properties included when running spark-submit.
# This is useful for setting default environmental settings.

# Example:
 spark.master                     spark://master:7077
# spark.eventLog.enabled           true
# spark.eventLog.dir               hdfs://namenode:8021/directory
# spark.serializer                 org.apache.spark.serializer.KryoSerializer
# spark.driver.memory              5g
# spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
~
~
```

Figure 2: Spark configuration - spark-defaults.conf.template file

The downloaded data sets are put onto the hdfs file system under `/data_mm16b029/` directory. This completes the initials setup for running the programs.



```
kpgudipaty@kpgudipaty-Inspiron-7460:~/Desktop/CS6847/assignment3$ hdfs dfs -put data_mm16b029/ /data_mm16b029
kpgudipaty@kpgudipaty-Inspiron-7460:~/Desktop/CS6847/assignment3$ hdfs dfs -ls /data_mm16b029
Found 3 items
-rw-r--r--   1 kpgudipaty supergroup      14351 2020-04-19 17:38 /data_mm16b029/ALS.txt
-rw-r--r--   1 kpgudipaty supergroup     302908 2020-04-19 17:38 /data_mm16b029/FP_Part-1.csv
-rw-r--r--   1 kpgudipaty supergroup   45580638 2020-04-19 17:38 /data_mm16b029/FP_Part-2.csv
kpgudipaty@kpgudipaty-Inspiron-7460:~/Desktop/CS6847/assignment3$
```

Figure 3: Datasets on hdfs system

ALS

The ALS algorithm provided by spark was run on the given dataset. The fine tuning was done for the range of iterations from 1 to 20. The regularization parameter was run for values $0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5$. The corresponding *RMSE* are recorded in `ALS_out.txt` file. The minimised *RMSE* for current experimental ranges was approx. 0.90144 achieved at *9* iterations and a *0.1* value for the regularization parameter. The default train-test split of 0.8-0.2 was used.

These values were used to minimize *RMSE* wrt to the train-test split of data. It was done from range 0.1 to 0.9 split ratio of training data with 0.1 increments. The minimum was achieved at 0.9 with an *RMSE* of approx. 0.89861. These can be found in `ALS_out_train_split.txt` file.

FP Growth

The FPGrowth example given by spark was used on both the datasets given. `FP_Part-1.csv` required just a bit of pre-pocessing. This preprocessing involved removing multiple instances of items in a transaction and to hold two-worded items together by joining them with an *underscore* between the words. For e.g. `french fries` became `french_fries`. The `FP_Part-2.csv` required additional data cleaning as well. The pre-processing and data cleaning used can be found in the `DataCleaner.py` file.

Five frequently occuring pairs are reported in `FP_out1.txt` and `FP_out2.txt`. The minimum support is 0.04 for `FP_Part-1.csv` and 0.02 for `FP_Part-2.csv`

****