Assignment 4

Return your solutions via Moodle page. We do not count belated returnings. You have to use .pdf file type for any written answers, and .scala for your code. Do not return any project or object files, etc. Mark your **full name and student number** clearly to all your solution files. Prepare to explain and discuss your solutions on Friday exercise session. Each exercise will be graded pass/fail. The purpose of this exercise set is to use spark cluster and learn to use spark advanced methods to optimize the performance of spark applications. Although We initially thought that we would provide more advanced exercises, now we can imagine that it will be very difficult and time consuming.

Exercise 1 (For python, we will have a separate exercise with GeoJSON, as magellan has issues with Python. We will add the new exercise soon by today.)

GeoJSON is a way to represent geographical information in a JSON format. The Magellan Spark package allows easy reading and operation on such data.
Download Countries of the world data from:
www.cs.helsinki.fi/u/lagerspe/countries-poly.geojson
And load it with http://geojson.io/
You should see some of the world's country boundaries.
Write the code to find the countries that contain these GPS coordinates. Note that these are in the Google Maps format, so the first one is in Finland, not Afghanistan:
60.2576009,24.9345427
40.7690327,-73.983803
39.9163488,116.3949606
51.4955324,-0.1407132
60.2039806,24.963352
-25.3456376,131.0283696
Use Magellan with spark-submit or spark shell as shown here
https://spark-packages.org/package/harsha2010/magellan
The examples here are helpful:
https://github.com/harsha2010/magellan
The answer to this exercise is the code that finds the countries that match these GPS points. After matching, list each GPS point together with the matching country name.

Update:- April 19th 2017 4:10 PM

Exercise for PySpark
An edited json data is added in the following URL,
https://raw.githubusercontent.com/shathil/BigDataExercises/master/week4/countries_and_coordinates.json

1. Now you can broadcast the following GPS points,
    60.2576009,24.9345427
    40.7690327,-73.983803
    39.9163488,116.3949606
    51.4955324,-0.1407132
    60.2039806,24.963352
    -25.3456376,131.0283696
2. Implement a Python function to check which country each points belongs to.
3. Use PySpark data frame or RDD or Row, whichever is convenient.

Note: You are free to implement any third party library you see fit or you also can implement your own logic to find a point is inside a polygon.

Hints:

1. You can also use PIP algorithm.
http://stackoverflow.com/questions/924171/geo-fencing-point-inside-outside-polygon

2.
http://stackoverflow.com/questions/217578/how-can-i-determine-whether-a-2d-point-is-within-a-polygon

Exercise 2

Use Carat-Context-Factor data and consider only the following features (cpuUsage, screenBrightness, wifiSginalStrength, batteryTemperature) and the dependent variable is the energyRate. Apply the following linear regression methods on this data set. Split the data set to two sets (train and test).

- LinearRegressionWithSGD
- RidgeRegressionWithSGD
- LassoWithSGD

Spark Example

http://spark.apache.org/docs/latest/mllib-linear-methods.html#linear-least-squares-lasso-and-ridge-regression

Spark Linear Regression Source code

https://github.com/apache/spark/blob/master/examples/src/main/scala/org/apache/spark/examples/mllib/LinearRegression.scala

https://github.com/apache/spark/tree/master/mllib/src/main/scala/org/apache/spark/mllib/regression/

Document the regression models and the configurations you applied, such as the step size and the number of iterations. Report the model and the error. In addition justify the selection of the configuration parameters (step size and the number of iterations) and discuss (criticize) the performance of these algorithms.

Exercise 3

Write the following spark applications to operate on a or a number streams

(1) You are given the address and port of a TCP stream source. Your program will receive a stream of integers within a batch interval of one second and compute the average of the numbers.
Server details:
Hostname ukko054
Port No 8890
Data Format number1,number2,....,number100
Note:- Data is of unicode 'utf-8' format, so you have to decode it after receiving it.

(2) You will assign five receivers, join the DStreams and find the duplicate numbers within a batch interval and keep a track of the total duplicate numbers (Hint stateful streaming).

(3) You will assign five receivers, join the DStreams and find the prime numbers within a batch interval.

Since the streams are not bounded, you need shut down the streaming context, after 2 minutes. Hint: streams can use timeout option.

Note:
A Python TCP Server that can generate 100 random numbers is added here. You can execute it locally too. It accepts a port number as command line argument. For example,

python3 tcp_random_number_gen.py 8050

If you wish to run the above file inside any ukko nodes, you have to manually edit the hostname(ukkoxxx instead of localhost) in the .py file.

Exercise 4: Distributed Matrix and Operations

This program requires the understanding of the partition related map functions. The program will have

1. Function to generate a distributed fat matrix, A, of size (1K x 10K) .
2. Then compute the distributed transpose of the matrix. The final transposed matrix, A', will be of size (10K x 1K).
3. Next, generate two corresponding block-wise matrixes containing a block of 1000x1000 elements. Matrix, A, will be a row matrix 1x10 blocks and A' will be a column matrix of 10x1 blocks.
4. Next multiply these two block matrixes and you will have a 1000x1000 matrix.

You are not allowed to use Spark mlib APIs.
http://www.math.nyu.edu/~neylon/linalgfall04/project1/dj/multpartmatrices.htm
Hint: check block matrix multiplication. The size of each matrix is 10000000 doubles. A single machine won't be enough to keep two such matrices in the memory. You must use more nodes in the ukko cluster. I have reduced the size of the matrices so that few machines are sufficient.

Exercise 5: Short notes

1. Compre HDFS and Tachyon in 15 sentences.
2. Compare Spark streaming and dataflow models in 15 sentences.