732A54 - Big Data Analytics

BDA1 Spark Exercises

Ashraf Sarhan (ashsa762)

December 23, 2016

1. What are the lowest and highest temperatures measured each year for the period 1950-2014. Provide the lists sorted in the descending order with respect to the maximum temperature. In this exercise you will use the `temperature-readings.csv` file.

- Non-parallelized program in Python to find the maximum temperatures for each year without using Spark. In this case you will run the program using: `python script.py`

serial_mmt_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 11 21:27:34 2016

@author: ashraf
"""
import time, argparse

parser = argparse.ArgumentParser(description='MinMaxTempExtractor')
parser.add_argument('-o', '--output', help='Output file', default='out_temp.csv')
requiredNamed = parser.add_argument_group('Required arguments')
requiredNamed.add_argument('-t', '--time', help='Time interval in years ex. 2005:2010', required=True)
requiredNamed.add_argument('-i', '--input', help='Input file', required=True)
args = parser.parse_args()
iFile = args.input
oFile = args.output
years =  args.time.split(":")
fromYear = int(years[0])
toYear = int(years[1])

start = time.time()
print('Running Python MinMaxTempExtractor:\nFrom %s To %s\nInput file: %s\nOutput file: %s'
      % (fromYear, toYear, iFile, oFile))
temp_dict = dict()
with open(iFile) as f:
    for l in f:
        line = l.split(";")
        year = int(line[1].split("-")[0])
        if year >= fromYear and year <= toYear:
            temp = temp_dict.get(year)
            station = line[0];
            curr_temp = float(line[3]);
```

```
            if not temp:
                #1st list for min temp and 2nd list for max temp
                temp_dict[year] = [[station, curr_temp], [station, curr_temp]]
            else:
                min = float(temp[0][1])
                max = float(temp[1][1])
                if curr_temp < min:
                    temp[0][0] = station
                    temp[0][1] = curr_temp
                if curr_temp > max:
                    temp[1][0] = station
                    temp[1][1] = curr_temp
#close the file after reading the lines.
f.close()

#sort temperatures descending by max temp
sorted_temp = temp_dict.items()
sorted_temp.sort(key=lambda x: x[1][1][1], reverse=True)

#write the output to file.
with open(oFile,'wb+') as f:
    for i in sorted_temp:
        #python will convert \n to os.linesep
        f.write('%s,%s,%s,%s,%s\n' % (i[0], i[1][0][0], i[1][0][1], i[1][1][0], i[1][1][1]))
#close the file after writting the lines.
f.close()
end = time.time()
print('Done in %s seconds' % (end - start))
```

spark_mmt_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
oFile = 'data/min_max_temperature'
fromYear = 1950
toYear = 2014

sc = SparkContext(appName="MinMaxTempExtractorSparkJob")

lines = sc.textFile(iFile)

lines = lines.map(lambda a: a.split(";"))

lines = lines.filter(lambda x: int(x[1][0:4]) >= 1950 and int(x[1][0:4]) <= 2014)
```

```
temperatures = lines.map(lambda x: (x[1][0:4], (x[0], float(x[3]))))

################## Custom MIN/MAX Function ############################

min = (lambda x, y: x if x[1] < y[1] else y)
max = (lambda x, y: x if x[1] > y[1] else y)

minTemperatures = temperatures.reduceByKey(min)

maxTemperatures = temperatures.reduceByKey(max)

minMaxTemp = minMaxTemp = minTemperatures.union(maxTemperatures). \
reduceByKey(lambda x,y: (x[0],x[1],y[0],y[1]))

sortedMinMaxTemp = minMaxTemp.sortBy(ascending=False, keyfunc=lambda a: a[1][3])

sortedMinMaxTempCsv = sortedMinMaxTemp.map(lambda a: '%s,%s,%s,%s,%s' \
% (a[0], a[1][0], a[1][1], a[1][2], a[1][3]))

sortedMinMaxTempCsv.coalesce(1).saveAsTextFile(oFile)
```

output:

```
year,station,min,station,max
1975,157860,-37.0,86200,36.1
1992,179960,-36.1,63600,35.4
1994,179960,-40.5,117160,34.7
2010,191910,-41.7,75250,34.4
2014,192840,-42.5,96560,34.4
1989,166870,-38.2,63050,33.9
1982,113410,-42.2,94050,33.8
1968,179950,-42.0,137100,33.7
1966,179950,-49.4,151640,33.5
1983,191900,-38.2,98210,33.3
```

- It's clearly appears that the Spark parallel version (1.5 min) takes much less time than the serial one (5 min) as this the main benefits of multiprocessing environment.

2. Count the number of readings for each month in the period of 1950-2014 which are higher than 10 degrees. Repeat the exercise, this time taking only distinct readings from each station. That is, if a station reported a reading above 10 degrees in some month, then it appears only once in the count for that month.

spark_temp_count_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
```

```
from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
oFile = 'data/over_ten_mth_temp_counts'
oFile2 = 'data/over_ten_temp_distinct_counts'
fromYear = 1950
toYear = 2014
target_temp = 10

sc = SparkContext(appName="TempCounterSparkJob")

lines = sc.textFile(iFile)

lines = lines.map(lambda a: a.split(";"))

lines = lines.filter(lambda x: int(x[1][0:4]) >= fromYear and int(x[1][0:4]) <= toYear \
and float(x[3]) > target_temp)

overTenMthTemp = lines.map(lambda x: (x[1][5:7], 1))

overTenMthTempCounts = overTenMthTemp.reduceByKey(lambda v1,v2: v1 + v2)

overTenMthTempCountsCsv = overTenMthTempCounts.map(lambda a: '%s,%s' % (a[0], a[1]))

overTenMthTempCountsCsv.coalesce(1).saveAsTextFile(oFile)

###################### Distinct Counting Per Station ######################
overTenStationTemp = lines.map(lambda x: (x[0], (x[1][5:7], float(x[3]))))

overTenStationTempDistinct = overTenStationTemp.distinct()

overTenStationTempDistinctCounts = overTenStationTempDistinct. \
map(lambda x: (x[0], 1)).reduceByKey(lambda v1,v2: v1 + v2)

overTenStationTempDistinctCountsCsv = overTenStationTempDistinctCounts. \
map(lambda a: '%s,%s' % (a[0], a[1]))

overTenStationTempDistinctCountsCsv.coalesce(1).saveAsTextFile(oFile2)
```

output:

```
station,count
72080,1137
133230,476
95160,1114
68560,821
83620,940
74240,1050
83170,837
103080,1117
134150,906
62140,758
```

3. Find the average monthly temperature for each available station in Sweden. Your result should include average temperature for each station for each month in the period of 1960-2014. Bear in mind that not every station has the readings for each month in this timeframe.

spark_temp_avg_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
oFile = 'data/station_avg_mth_temp'
fromYear = 1960
toYear = 2014

sc = SparkContext(appName="AvgTempSparkJob")

lines = sc.textFile(iFile)

lines = lines.map(lambda a: a.split(";"))

lines = lines.filter(lambda x: int(x[1][0:4]) >= fromYear and int(x[1][0:4]) <= toYear)

temperatures = lines.map(lambda x: (x[0]+','+x[1][5:7], (float(x[3]), 1)))

stationMthTemp = temperatures.reduceByKey(lambda v1,v2: (v1[0]+v2[0], v1[1]+v2[1]))

stationAvgMthTemp = stationMthTemp.map(lambda a: (a[0], a[1][0]/a[1][1]))

stationAvgMthTempCsv = stationAvgMthTemp.map(lambda a: '%s,%s' % (a[0], a[1]))

stationAvgMthTempCsv.coalesce(1).saveAsTextFile(oFile)
```

output:

```
station,month,avgTemp
134100,11,-2.29005847953
115220,06,13.0421685863
155710,02,-9.1140943194
54990,05,12.1786961583
138400,01,-8.6290626902
96000,01,-2.4259727303
83170,03,0.343137254902
158850,02,-11.0974036317
104300,09,8.7858941528
77180,09,12.7023285199
```

4. Provide a list of stations with their associated maximum measured temperatures and maximum measured daily precipitation. Show only those stations where the maximum temperature is between 25 and 30 degrees and maximum daily precipitation is between 100 mm and 200 mm.

spark_max_temp_prec_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
iFile2 = 'data/precipitation-readings.csv'
oFile = 'data/max_temperature_precipitation'

sc = SparkContext(appName="MaxTempPrecExtractorSparkJob")

######### Max Temperatures #########
temperatures = sc.textFile(iFile)
temperatures = temperatures.map(lambda a: a.split(";"))
temperatures = temperatures.map(lambda x: (x[0], float(x[3])))
maxTemperatures = temperatures.reduceByKey(max)
maxTemperatures = maxTemperatures.filter(lambda a: a[1] > 25 and a[1] < 30)

######### Max Precipitations #########
precipitations = sc.textFile(iFile2)
precipitations = precipitations.map(lambda a: a.split(";"))
precipitations = precipitations.map(lambda x: (x[0]+','+x[1], float(x[3])))
maxPrecipitations = precipitations.reduceByKey(max)
maxPrecipitations = maxPrecipitations.filter(lambda a: a[1] > 100 and a[1] < 200) \
.map(lambda x: (x[0].split(",")[0], x[1]))

######### Merged Max Temperatures/Precipitations #########
maxTempPrec = maxTemperatures.union(maxPrecipitations)

maxTempPrec.coalesce(1).saveAsTextFile(oFile)
```

output:

```
81060,28.0
83620,29.4
133230,28.2
134150,29.7
62140,27.7
132170,28.0
89240,29.9
132620,26.8
156730,29.1
```

```
78280,28.7
```

5. Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file). In order to do this, you will first need to calculate the total monthly precipitation for each station before calculating the monthly average (by averaging over stations).

spark_ostergot_avg_prec_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext

iFile = 'data/stations-Ostergotland.csv'
iFile2 = 'data/precipitation-readings.csv'
oFile = 'data/OstergotlandAveMonthlyPrec'

sc = SparkContext(appName="OstergotlandAvgMonthlyPrecSparkJob")

ostergotlandStations = sc.textFile(iFile)

ostergotlandStations = ostergotlandStations.map(lambda line: line.split(";")) \
.map(lambda x: int(x[0])).distinct().collect()

isOstergotlandStation = (lambda s: s in ostergotlandStations)

precipitations = sc.textFile(iFile2)

daily_precipitations = precipitations.map(lambda line: line.split(";")) \
.filter(lambda x: isOstergotlandStation(int(x[0])))

daily_precipitations = daily_precipitations.map(lambda x: (x[0]+','+x[1], float(x[3]))). \
reduceByKey(lambda a, b: a + b)

monthly_precipitations = daily_precipitations. \
map(lambda x:(x[0].split("-")[0]+','+x[0].split("-")[1], (x[1], 1)))

monthly_precipitations = monthly_precipitations. \
reduceByKey(lambda v1, v2: (v1[0] + v2[0], v1[1] + v2[1]))

monthly_precipitations = monthly_precipitations.map(lambda x: (x[0], x[1][0] / x[1][1]))

monthly_precipitations_csv = monthly_precipitations.map(lambda a: '%s,%s' % (a[0], a[1]))

monthly_precipitations_csv.coalesce(1).saveAsTextFile(oFile)
```

output:

```
station,year,month,avgPrec
85460,1996,04,0.506666666667
75520,2004,12,0.648387096774
85050,2015,02,0.757142857143
86420,2014,07,0.133333333333
75520,1997,05,1.26129032258
75520,2006,12,0.887096774194
75520,2008,01,1.20967741935
75520,2008,06,0.816666666667
85460,2013,03,0.109677419355
87140,2000,06,1.68666666667
```

6. Compare the average monthly temperature (find the difference) in the period 1950-2014 for all stations in Östergotland with long-term monthly averages in the period of 1950-1980.

spark_ostergot_avg_temp_diff_extractor.py:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 12 17:06:24 2016

@author: ashraf
"""
from pyspark import SparkContext

iFile = 'data/stations-Ostergotland.csv'
iFile2 = 'data/temperature-readings.csv'
oFile = 'data/OstergotlandAvgMonthlyDiffTemp'

sc = SparkContext(appName="OstergotlandAvgMonthlyTempDiffSparkJob")

ostergotlandStations = sc.textFile(iFile)

ostergotlandStations = ostergotlandStations.map(lambda line: line.split(";")). \
map(lambda x: int(x[0])).distinct().collect()

isOstergotlandStation = (lambda s: s in ostergotlandStations)

temperatures = sc.textFile(iFile2)

temperatures = temperatures.map(lambda line: line.split(";")). \
filter(lambda x: isOstergotlandStation(int(x[0])) and \
int(x[1][0:4]) >= 1950 and int(x[1][0:4]) <= 2014)

daily_temperatures = temperatures.map(lambda x: (x[1], (float(x[3]), float(x[3])))). \
reduceByKey(lambda t1, t2: (min(t1[0], t2[0]), max(t1[1], t2[1])))

monthly_temperatures = daily_temperatures.map(lambda x:(x[0].split("-")[0]+', \
'+x[0].split("-")[1], (x[1][0]+x[1][1], 2)))
```

```
monthly_temperatures = monthly_temperatures. \
reduceByKey(lambda v1, v2: (v1[0] + v2[0], v1[1] + v2[1]))

monthly_avg_temperatures = monthly_temperatures.map(lambda x: (x[0], x[1][0] / x[1][1]))

longterm_monthly_avg_temperatures = monthly_avg_temperatures. \
filter(lambda x: int(x[0].split(",")[0]) >= 1950 and int(x[0].split(",")[0]) <= 1980). \
map(lambda x: (x[0].split(",")[1], (x[1], 1))). \
reduceByKey(lambda t1, t2:(t1[0] + t2[0], t1[1] + t2[1])).map(lambda t: (t[0], t[1][0]/t[1][1]))

longTermAvgTempLookup = {month: temp for month, temp in longterm_monthly_avg_temperatures.collect()}

final_res = monthly_avg_temperatures.map(lambda x: (x[0], x[1] - \
longTermAvgTempLookup[x[0].split(",")[1]]))

monthly_avg_diff_temperatures_csv = final_res.map(lambda a: '%s,%s' % (a[0], a[1]))

monthly_avg_diff_temperatures_csv.coalesce(1).saveAsTextFile(oFile)
```

output:

```
year,month,avgDiffPrec
1992,06,0.611666666667
1967,11,1.51274193548
1993,09,-3.8173655914
1970,02,-7.51101223582
1998,04,-1.05983870968
1966,01,-3.92185223725
2001,06,-2.445
2002,07,0.460926118626
1971,12,2.77164412071
1991,07,0.851248699272
```