L1-jiawu449-zijfe244

Jiawei Wu & Zijie Feng

2019/5/25

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.

- a) Extend the program to include the station number (not the station name) where the maximum/minimum temperature was measured.

```
from pyspark import SparkContext

# sc.stop()
def max_temperature(a,b):
    if a>=b:
        return a
    else:
        return b
def min_temperature(a,b):
    if a>=b:
        return b
    else:
        return a
sc = SparkContext(appName="exercise")

# read csv file
temperature_file=sc.textFile("/user/x_zijfe/data1/temperature-readings.csv")
lines = temperature_file.map(lambda line:line.split(";"))
# print(*lines.take(1000), sep='\n')

# all the temperatures from 1950 to 2014
# make 'station number' and 'temperature' as a multi-value
year_temperature = lines.map(lambda x:(x[1][0:4],(float(x[3]),int(x[0]))))
year_temperature = year_temperature.filter(lambda x: int(x[0])>=1950 and int(x[0])<=2014)

# reduce 'max_temperature' by key 'year', it will also work on value 'stations number'
max_temperature = year_temperature.reduceByKey(max_temperature)
max_temperatureSorted = max_temperature.sortBy(ascending = False, keyfunc=lambda k:k[1])
# print(*max_temperatureSorted.take(64),sep="\n")
max_temperatureSorted.saveAsTextFile("./results/Q1a/max_temperatureSorted")

min_temperature = year_temperature.reduceByKey(min_temperature)
min_temperatureSorted = min_temperature.sortBy(ascending = False, keyfunc=lambda k:k[1])
# print(*min_temperatureSorted.take(64),sep="\n")
min_temperatureSorted.saveAsTextFile("./results/Q1a/min_temperatureSorted")
```

```
# Head part max_temperatureSorted
(u'1975', (36.1, 86200))
(u'1992', (35.4, 63600))
```

```
(u'1994', (34.7, 117160))
(u'2014', (34.4, 96560))
(u'2010', (34.4, 75250))
(u'1989', (33.9, 63050))
(u'1982', (33.8, 94050))
(u'1968', (33.7, 137100))
(u'1966', (33.5, 151640))
(u'1983', (33.3, 98210))
(u'2002', (33.3, 78290))
(u'1970', (33.2, 103080))
(u'1986', (33.2, 76470))
(u'1956', (33.0, 145340))
(u'2000', (33.0, 62400))
(u'1959', (32.8, 65160))
(u'1991', (32.7, 137040))
(u'2006', (32.7, 75240))
(u'1988', (32.6, 102540))
(u'2011', (32.5, 172770))
(u'1999', (32.4, 98210))
(u'2003', (32.2, 136420))
(u'2008', (32.2, 102390))
(u'1955', (32.2, 97260))
(u'2007', (32.2, 86420))
(u'1973', (32.2, 71470))
(u'1953', (32.2, 65160))
(u'2005', (32.1, 107140))
(u'1969', (32.0, 97260))
(u'1979', (32.0, 63600))
...

# End part min_temperatureSorted
...
(u'1963', (-41.0, 181900))
(u'1955', (-41.2, 160790))
(u'1969', (-41.5, 181900))
(u'2003', (-41.5, 179960))
(u'2010', (-41.7, 191910))
(u'1996', (-41.7, 155790))
(u'1962', (-42.0, 181900))
(u'2011', (-42.0, 179960))
(u'1968', (-42.0, 179950))
(u'1951', (-42.0, 155910))
(u'1950', (-42.0, 155910))
(u'1976', (-42.2, 192830))
(u'2002', (-42.2, 169860))
(u'1982', (-42.2, 113410))
(u'2014', (-42.5, 192840))
(u'1977', (-42.5, 179950))
(u'2012', (-42.7, 191910))
(u'1998', (-42.7, 169860))
(u'1958', (-43.0, 159970))
(u'1985', (-43.4, 159970))
(u'1959', (-43.6, 159970))
```

```
(u'1965', (-44.0, 189780))
(u'1981', (-44.0, 166870))
(u'2001', (-44.0, 112530))
(u'1979', (-44.0, 112170))
(u'1986', (-44.2, 167860))
(u'1971', (-44.3, 166870))
(u'1980', (-45.0, 191900))
(u'1956', (-45.0, 160790))
(u'1967', (-45.4, 166870))
(u'1987', (-47.3, 123480))
(u'1978', (-47.7, 155940))
(u'1999', (-49.0, 192830))
(u'1966', (-49.4, 179950))
```

- b) (not for the SparkSQL lab) Write the 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

This program will read the local file (not from HDFS). The local file is available under

/nfshome/hadoop_examples/shared_data/temperatures-big.csv.

How does the runtime compare to the Spark version? Use logging (add the –conf spark.eventLog.enabled=true flag) to check the execution of the Spark program. Repeat the exercise, this time using *temperatures-big.csv* file available on hdfs. Explain the differences and try to reason why such runtimes were observed.

```python
import time
import io
start = time.time()

max_temp = {year:-99 for year in range(1950, 2015)}
with io.open("/nfshome/hadoop_examples/shared_data/temperatures-big.csv","r") as csvfile:
    for row in csvfile:
        reading = row.split(";")
        year = int(reading[1][0:4])
        if year >= 1950 and year <= 2014:
            temp = float(reading[3])
#            station = float(reading[0])
            if temp > max_temp[year]:
                max_temp[year] = temp

max_temp_sorted = sorted(max_temp.items(), key=lambda x: x[1], reverse=True)
# print(*max_temp_sorted, sep="\n")

end = time.time()
final_time = time.strftime('%H:%M:%S', time.gmtime(end-start))
print ("RUNTIME = " + final_time + "  #######################  RUNTIME IS HERE!!!  #################")

from pyspark import SparkContext
sc = SparkContext(appName="exercise")
rdd = sc.parallelize(max_temp_sorted)
# print(*rdd.take(20),sep="\n")
rdd.saveAsTextFile("./results1/1q1b")
```

Our non-parallel python code does not work for *heffa local* file. It will throw the Java TimeoutException:

Futures timed out after [100000 milliseconds]. When we use *hdfs* path, the `io.open` cannot read the csv file. In addition, we cannot save file in HDFS as well, so we change our results to rdd and use saveAsTextFile to save data.

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.

In this exercise you will use the *temperature-readings.csv* file. The output should contain the following information:

Year, month, count

```
### Q2
from pyspark import SparkContext
# sc.stop()
sc = SparkContext(appName="exercise")

# read csv file
temperature_file=sc.textFile("temperature-readings-tiny.csv")
lines = temperature_file.map(lambda line:line.split(";"))

# map necessary keys and values
# (year-month, temperature), stationNumber
year_temperature = lines.map(lambda x:(x[1][0:7],float(x[3]),x[0]))
year_temperature = year_temperature.filter(lambda x: int(x[0][0:4])>=1950 \
                                           and int(x[0][0:4])<=2014 and int(x[1])>=10)
countsource1 = year_temperature.map(lambda x:(tuple([x[0],x[2]]),1))

# if a station reported a reading above 10 degrees in some month,
# then it appears only once in the count for that month.
unique1 = countsource1.distinct()

# year, month, #readings
countsource2 = unique1.map(lambda x:(x[0][0],1))
counts2 = countsource2.reduceByKey(lambda v1,v2:v1+v2)
counts = counts2.map(lambda x:(x[0][0:4],x[0][5:7],x[1]))
counts = counts.sortBy(lambda x: x[2], ascending = False)
# print(*counts.take(20),sep="\n")
counts.saveAsTextFile("counts")
```

```
# Head part
(u'1972', u'10', 378)
(u'1973', u'06', 377)
(u'1973', u'05', 377)
(u'1972', u'05', 376)
(u'1973', u'09', 376)
(u'1972', u'08', 376)
(u'1971', u'08', 375)
(u'1972', u'06', 375)
(u'1972', u'09', 375)
(u'1972', u'07', 374)
(u'1971', u'09', 374)
(u'1971', u'06', 374)
(u'1973', u'08', 373)
(u'1971', u'05', 373)
(u'1974', u'08', 372)
(u'1974', u'06', 372)
```

```
(u'1974', u'05', 370)
(u'1973', u'07', 370)
(u'1974', u'09', 370)
(u'1970', u'08', 370)
(u'1971', u'07', 370)
(u'1970', u'09', 369)
(u'1970', u'06', 369)
(u'1975', u'09', 369)
(u'1976', u'05', 369)
(u'1975', u'06', 368)
(u'1976', u'06', 368)
(u'1975', u'08', 367)
(u'1975', u'05', 367)
(u'1970', u'05', 366)
(u'1976', u'09', 365)
(u'1977', u'06', 364)
(u'1967', u'05', 363)
(u'1976', u'08', 363)

...
```

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. In this exercise you will use the *temperature-readings.csv* file.

The output should contain the following information:

Year, month, station number, average monthly temperature

```
###Q3 final
#sc.stop()
from pyspark import SparkContext
sc = SparkContext(appName="exercise")

# read csv file
temperature_file=sc.textFile("/user/x_zijfe/data1/temperature-readings.csv")
lines = temperature_file.map(lambda line:line.split(";"))

# map necessary keys and values
# (year-month, temperature), stationNumber
year_temperature = lines.map(lambda x:(x[1][0:7],float(x[3]),x[0]))
year_temperature = year_temperature.filter(lambda x: int(x[0][0:4])>=1960 and int(x[0][0:4])<=2014)

# (year-month, stationNumber), #readings
countSource = year_temperature.map(lambda x:((x[0],x[2]),1))
counts=countSource.reduceByKey(lambda x,y: x+y)

# (year-month, stationNumber), tempSum
temperature = year_temperature.map(lambda x:((x[0],x[2]),x[1]))
tempSum=temperature.reduceByKey(lambda v1,v2:v1+v2)

# (year, month, stationNumber, TempAve)
aveSource = tempSum.join(counts)
aveTemp = aveSource.map(lambda x:(x[0][0][0:4],x[0][0][5:7],x[0][1],round(x[1][0]/x[1][1],2)))
aveTemp = aveTemp.sortBy(lambda x: x[3], ascending = False)

# print(*aveTemp.take(20),sep="\n")
averagetemp.saveAsTextFile("./results1/1q3")
```

```
# End part
(u'1985', u'02', u'167860', -23.07)
(u'1966', u'02', u'192710', -23.07)
(u'1968', u'01', u'169930', -23.09)
(u'1985', u'12', u'156940', -23.09)
(u'1985', u'02', u'162980', -23.14)
(u'1966', u'01', u'181900', -23.17)
(u'1967', u'01', u'181900', -23.19)
(u'1985', u'02', u'147570', -23.25)
(u'1966', u'02', u'166810', -23.28)
(u'1985', u'02', u'191900', -23.35)
(u'1987', u'01', u'181900', -23.39)
(u'1987', u'01', u'169880', -23.42)
(u'1987', u'01', u'170790', -23.43)
(u'1985', u'02', u'172790', -23.47)
```

```
(u'1966', u'02', u'179950', -23.53)
(u'1985', u'02', u'160890', -23.6)
(u'1985', u'02', u'167980', -23.61)
(u'1985', u'02', u'182930', -23.63)
(u'1966', u'02', u'191900', -23.63)
(u'1985', u'02', u'166810', -23.64)
(u'1985', u'02', u'179950', -23.64)
(u'1985', u'02', u'183980', -23.7)
(u'1993', u'12', u'166900', -23.8)
(u'1985', u'02', u'159970', -23.96)
(u'1966', u'02', u'189780', -23.98)
(u'1966', u'02', u'192830', -24.03)
(u'1966', u'02', u'181900', -24.09)
(u'1966', u'02', u'166870', -24.23)
(u'1985', u'02', u'183760', -24.48)
(u'1966', u'02', u'159970', -24.69)
(u'1966', u'02', u'167860', -24.73)
(u'1985', u'02', u'166870', -24.77)
(u'1985', u'02', u'169880', -25.41)
(u'1985', u'02', u'192830', -26.19)
(u'1985', u'02', u'181900', -26.67)
```

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. In this exercise you will use the *temperature-readings.csv* and *precipitation-readings.csv* files.

The output should contain the following information:

Station number, maximum measured temperature, maximum daily precipitation

HINT: The correct result for this question should be empty.

```
###Q4 final
from pyspark import SparkContext

sc = SparkContext(appName="exercise")

precipitation_file=sc.textFile("/user/x_zijfe/data1/precipitation-readings.csv")
linesp = precipitation_file.map(lambda line:line.split(";"))
temperature_file=sc.textFile("/user/x_zijfe/data1/temperature-readings.csv")
linest = temperature_file.map(lambda line:line.split(";"))

# max temperature
temperatures = linest.map( lambda x:(str(x[0]),float(x[3])) )
stations_temperatures = temperatures.reduceByKey(lambda x,y: x if x>y else y)

# max measured precipitation
precipitations = linesp.map(lambda x:((x[0],x[1]),float(x[3])))
daily_precipitations = precipitations.reduceByKey(lambda x,y: x+y)
daily_precipitations = daily_precipitations.map(lambda x:(x[0][0],x[1]))
stations_precipitations = daily_precipitations.reduceByKey(lambda x,y: x if x>y else y)
stations = stations_precipitations.join(stations_temperatures)

stations = stations.filter(lambda x: x[1][0]>=100 and x[1][0]<=200 and x[1][1]>=25 and x[1][1]<=30)
stations.saveAsTextFile("./results1/1q4")
```

The output is nothing.

5)

Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file) for the period 1993-2016. 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).

In this exercise you will use the *precipitation-readings.csv* and *stations-Ostergotland.csv* files. HINT (not for the SparkSQL lab): Avoid using joins here!*stations-Ostergotland.csv* is small and if distributed will cause a number of unnecessary shuffles when joined with precipitation RDD. If you distribute precipitation-readings.csv then either repartition your stations RDD to 1 partition or make use of the collect to acquire a python list and broadcast function to broadcast the list to all nodes.

The output should contain the following information:

Year, month, average monthly precipitation

```
from pyspark import SparkContext
# sc.stop()
sc = SparkContext(appName="exercise")

precipitation_file=sc.textFile("/user/x_zijfe/data1/precipitation-readings.csv")
linesp = precipitation_file.map(lambda line:line.split(";"))

stations_Ostergotland=sc.textFile("/user/x_zijfe/data1/stations-Ostergotland.csv")
liness = stations_Ostergotland.map(lambda line:line.split(";"))

######## the list of all the stations
stations = liness.map(lambda x:x[0]).distinct().collect()   # by collect(), output is a list
# print(*stations)

######## the daily-temperatures we need for each stations in list
daily_precipitation = linesp.map(lambda x:((x[1][0:7],x[0]),float(x[3])))
daily_precipitation = daily_precipitation.filter(lambda x: int(x[0][0][0:4])>=1996 \
                                                  and int(x[0][0][0:4])<=2016 and str(x[0][1]) in statio

######## the month-temperatures we need for each stations
month_precipitationSum = daily_precipitation.reduceByKey(lambda v1,v2:v1+v2)
month_precipitationSum = month_precipitationSum.map(lambda x:(x[0][1],(x[0][0],x[1])) )

monthPrecValue = month_precipitationSum.map(lambda x:(x[1]))
monthPrecValue = monthPrecValue.reduceByKey(lambda v1,v2:v1+v2)

########## # of date
dates = month_precipitationSum.map(lambda x:(x[1][0],1))
dates = dates.reduceByKey(lambda v1,v2:v1+v2)

########## average
monthAve = monthPrecValue.union(dates).reduceByKey(lambda v1,v2:v1/v2)
monthAve = monthAve.map(lambda x: (int(x[0][0:4]),int(x[0][5:7]),round(x[1],2)))
monthAve = monthAve.sortBy(lambda x: (x[0],[1]), ascending = False)

# print(monthAve.take(10))
monthAve.saveAsTextFile("./results1/1q5")
```

```
# Head part
(u'2016', u'06', 47.66)
(u'2016', u'01', 22.32)
```

```
(u'2016', u'05', 29.25)
(u'2016', u'07', 0.0)
(u'2016', u'02', 21.56)
(u'2016', u'04', 26.9)
(u'2016', u'03', 19.96)
(u'2015', u'05', 93.22)
(u'2015', u'11', 63.89)
(u'2015', u'02', 24.82)
(u'2015', u'08', 26.99)
(u'2015', u'12', 28.93)
(u'2015', u'04', 15.34)
(u'2015', u'01', 59.11)
(u'2015', u'10', 2.26)
(u'2015', u'06', 78.66)
(u'2015', u'03', 42.61)
(u'2015', u'09', 101.3)
(u'2015', u'07', 119.1)
(u'2014', u'05', 58.0)
(u'2014', u'12', 35.46)
(u'2014', u'09', 48.45)
(u'2014', u'04', 31.76)
(u'2014', u'03', 36.56)
(u'2014', u'01', 62.58)
(u'2014', u'08', 90.81)
(u'2014', u'07', 22.99)
(u'2014', u'02', 43.71)
(u'2014', u'10', 72.14)
(u'2014', u'06', 75.14)
(u'2014', u'11', 52.43)
...
```

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. Make a plot of your results.

HINT: The first step is to find the monthly averages for each station. In the next step, you can average over all stations to acquire the average temperature for a specific year and month. This RDD/Data Frame can be used to compute the long-term average by averaging over all the years in theinterval.

The output should contain the following information:

Year, month, difference

```
from pyspark import SparkContext
# import matplotlib.pyplot as plt

# sc.stop()
sc = SparkContext(appName="exercise")
file=sc.textFile("/user/x_zijfe/data1/temperature-readings.csv")
linesp = file.map(lambda line:line.split(";"))

stations_Ostergotland=sc.textFile("/user/x_zijfe/data1/stations-Ostergotland.csv")
liness = stations_Ostergotland.map(lambda line:line.split(";"))

######## the list of all the stations
stations = liness.map(lambda x:x[0]).distinct().collect()

################ step 1 ###############
daily_temp = linesp.map(lambda x:((x[1][0:7],x[0]),float(x[3])))
daily_temp = daily_temp.filter(lambda x: str(x[0][1]) in stations)
month_temp_sum = daily_temp.reduceByKey(lambda v1,v2:v1+v2)
# print(*month_temp_sum.take(10),sep="\n")

num_month = daily_temp.map(lambda x:(x[0], 1))
num_month = num_month.reduceByKey(lambda v1,v2:v1+v2)
# print(*num_month.take(10), sep="\n")
month_temp_ave = month_temp_sum.union(num_month).reduceByKey(lambda v1,v2:v1/v2)
# print(*month_temp_ave.take(10),sep="\n")

################ step 2 ###############
month_temp_ave2 = month_temp_ave.map(lambda x: (x[0][0], x[1]))
month_temp_ave2 = month_temp_ave2.reduceByKey(lambda v1,v2: v1+v2)
num_station = month_temp_ave.map(lambda x: (x[0][0], 1))
num_station = num_station.reduceByKey(lambda v1,v2:v1+v2)
month_temp_ave2 = month_temp_ave2.union(num_station).reduceByKey(lambda v1,v2:v1/v2)
month_temp_ave2 = month_temp_ave2.map(lambda x: [int(x[0][0:4]), int(x[0][5:7]), x[1]])
# print(*month_temp_ave2.take(10),sep="\n")

############### final #################
# list[year, month, value]
ave1 = month_temp_ave2.filter(lambda x: x[0]>=1950 and x[0]<=2014)\
                      .sortBy(lambda x: (x[0],[1]), ascending = False)
final = ave1.collect()
# dict(month: value)
ave2 = month_temp_ave2.filter(lambda x: x[0]>=1950 and x[0]<=1980)\
                      .sortBy(lambda x: (x[0],[1]), ascending = False)
```

```
sum2 = ave2.map(lambda x: (x[1], x[2])).reduceByKey(lambda v1, v2: v1+v2)
num2 = ave2.map(lambda x: (x[1], 1)).reduceByKey(lambda v1, v2: v1+v2)
ave2 = dict(sum2.union(num2).reduceByKey(lambda v1,v2:v1/v2).collect())

for i in range(len(final)):
    month = final[i][1]
    final[i][2] -= ave2[month]

# We change the rdd to list for calculation.
# Since we cannot save list directly, we have
# to change from list to DF, and then to rdd for saving.
lll = sc.parallelize(final)
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df = sqlContext.createDataFrame(lll)
df = df.sort(df._1.desc(), df._2.desc()).withColumnRenamed("_1","year")\
                              .withColumnRenamed("_2","month")\
                              .withColumnRenamed("_3","difference")
fff = df.rdd
fff.saveAsTextFile("./results1/1q6")
```

```
# Head part (differences from 2012 to 2014)
Row(year=2014, month=12, difference=0.8090050128011185)
Row(year=2014, month=11, difference=2.095774582278379)
Row(year=2014, month=10, difference=1.536984352557103)
Row(year=2014, month=9, difference=0.045288688362905556)
Row(year=2014, month=8, difference=-0.7553156355884578)
Row(year=2014, month=7, difference=2.168129032266787)
Row(year=2014, month=6, difference=-1.878825133086833)
Row(year=2014, month=5, difference=0.15765101764698564)
Row(year=2014, month=4, difference=2.121103686922506)
Row(year=2014, month=3, difference=4.222140505337416)
Row(year=2014, month=2, difference=5.25560805504952)
Row(year=2014, month=1, difference=0.9193686872017368)
Row(year=2013, month=12, difference=3.8778419122262773)
Row(year=2013, month=11, difference=0.9942484465096872)
Row(year=2013, month=10, difference=0.6741854602767257)
Row(year=2013, month=9, difference=-1.0152003851953104)
Row(year=2013, month=8, difference=-0.3856878368397414)
Row(year=2013, month=7, difference=0.14340585614449708)
Row(year=2013, month=6, difference=-0.6412691757865598)
Row(year=2013, month=5, difference=1.4171806693274611)
Row(year=2013, month=4, difference=-0.7561818989841251)
Row(year=2013, month=3, difference=-3.5385474271177197)
Row(year=2013, month=2, difference=0.6025816048169408)
Row(year=2013, month=1, difference=-0.6509672815448826)
Row(year=2012, month=12, difference=-3.5723414709038566)
Row(year=2012, month=11, difference=1.3384372851086859)
Row(year=2012, month=10, difference=-1.4712725765528818)
Row(year=2012, month=9, difference=-0.4785962317840351)
Row(year=2012, month=8, difference=-0.8105206956571269)
Row(year=2012, month=7, difference=-0.7777453374623384)
Row(year=2012, month=6, difference=-3.1633179928387314)
Row(year=2012, month=5, difference=0.7109398802281408)
Row(year=2012, month=4, difference=-0.5505924661981494)
Row(year=2012, month=3, difference=4.352904131814594)
Row(year=2012, month=2, difference=0.06863220126275404)
Row(year=2012, month=1, difference=1.482400932050769)
...
```