BIG DATA LAB 1

Rabnawaz Jansher & Saman Zahid

7 May 2018

BDA1 -Spark -Exercises

Question 1

Part A

```
from pyspark import SparkContext
# function for calculating max temperature
# also returns associated stations
def max_temperature(a,b):
    if a[0]>=b[0]:
        return a
    else:
        return b
sc = SparkContext(appName = "lab1-q1")
# reading data
temperature_file = sc.textFile("/user/x_samza/data/temperature-readings.csv")
# splitting columns
lines = temperature_file.map(lambda line: line.split(";"))
# making key value pair keeping year as key and temp and station as values
station_year_temperature = lines.map(lambda x:
(x[1][0:4],(float(x[3]),x[0])))
# filtering data between 1950 and 2014
station_year_temperature = station_year_temperature.filter(lambda x:
int(x[0]) >= 1950 and int(x[0]) <= 2014)
# finding max_temperature based on key - 'year'
max_temperatures = station_year_temperature.reduceByKey(lambda
(x11,x21),(x12,x22): max_temperature((x11,x21),(x12,x22)))
# sorting result in descending order of max temperature
max_temperatureSorted = max_temperatures.sortBy(ascending = False , keyfunc =
lambda k : k[1][0])
# saving result to hadoop
max_temperatureSorted.saveAsTextFile("max_temperature_station")
```

part-00000

```
(u'1975', (36.1, u'86200'))
(u'1992', (35.4, u'63600'))
(u'1994', (34.7, u'117160'))
(u'2014', (34.4, u'96560'))
(u'2010', (34.4, u'75250'))
(u'1989', (33.9, u'63050'))
(u'1982', (33.8, u'94050'))
(u'1968', (33.7, u'137100'))
(u'1966', (33.5, u'151640'))
(u'2002', (33.3, u'78290'))
(u'1983', (33.3, u'98210'))
(u'1986', (33.2, u'76470'))
(u'1970', (33.2, u'103080'))
(u'1956', (33.0, u'145340'))
(u'2000', (33.0, u'62400'))
(u'1959', (32.8, u'65160'))
(u'1991', (32.7, u'137040'))
(u'2006', (32.7, u'75240'))
(u'1988', (32.6, u'102540'))
(u'2011', (32.5, u'172770'))
(u'1999', (32.4, u'98210'))
(u'1955', (32.2, u'97260'))
(u'2007', (32.2, u'86420'))
(u'1973', (32.2, u'71470'))
(u'2003', (32.2, u'136420'))
(u'2008', (32.2, u'102390'))
(u'1953', (32.2, u'65160'))
(u'2005', (32.1, u'107140'))
(u'1979', (32.0, u'63600'))
(u'1969', (32.0, u'71470'))
(u'2001', (31.9, u'62400'))
(u'1997', (31.8, u'74180'))
(u'1977', (31.8, u'94180'))
(u'2013', (31.6, u'98210'))
(u'2009', (31.5, u'81370'))
(u'2012', (31.3, u'54990'))
(u'1971', (31.2, u'65130'))
(u'1964', (31.2, u'76430'))
(u'1972', (31.2, u'137100'))
(u'1976', (31.1, u'75040'))
```

Part B

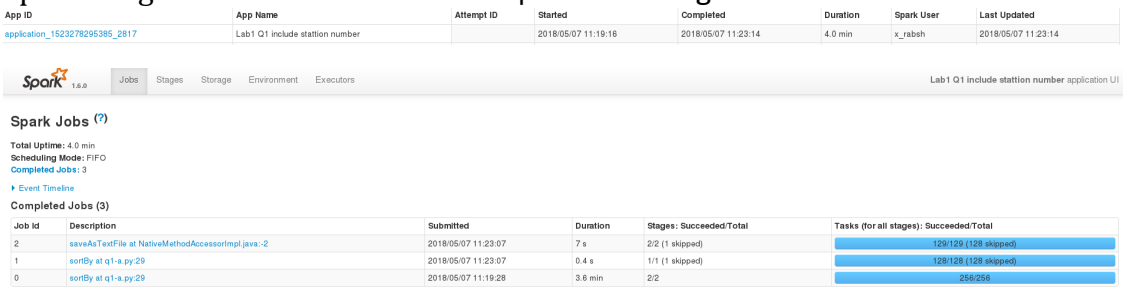
```
#non parallel python program
#import csv
import csv
import time
start_time = time.time()
#dict
data = {}
# read data
with open('/nfshome/hadoop_examples/shared_data/temperatures-big.csv') as
csvDataFile:
    #split data
    csvReader = csv.reader(csvDataFile,delimiter=';')
    for row in csvReader:
        year = int(row[1][0:4])
        #filter data and compare temperature
        if int(year) >= 1950 and int(year) <= 2014:
            temp = float(row[3])
            if not data:
                data[year] = temp
            else:
                if year in data.keys():
                    if data[year] < temp:
                        data[year] = temp
                else:
                    data[year] = temp

#sort data
for row in sorted(data.items(),key=lambda x: (x[1],x[0]),reverse=True):
  print row
#print time
print("--- %s seconds ---" % (time.time() - start_time))
```

q1b.py

```
(2000, 33.0)
(1956, 33.0)
(1959, 32.8)
(2006, 32.7)
(1991, 32.7)
(1988, 32.6)
(2011, 32.5)
(1999, 32.4)
(2008, 32.2)
(2007, 32.2)
(2003, 32.2)
(1973, 32.2)
(1955, 32.2)
(1953, 32.2)
(2005, 32.1)
(1979, 32.0)
(1969, 32.0)
(2001, 31.9)
(1997, 31.8)
(1977, 31.8)
(2013, 31.6)
(2009, 31.5)
(2012, 31.3)
(1972, 31.2)
(1971, 31.2)
(1964, 31.2)
(1976, 31.1)
(1963, 31.0)
(1961, 31.0)
(1996, 30.8)
(1995, 30.8)
(1978, 30.8)
(1958, 30.8)
(1974, 30.6)
(1954, 30.5)
(1980, 30.4)
(1952, 30.4)
(2004, 30.2)
(1990, 30.2)
(1985, 29.8)
(1957, 29.8)
```

comparison

Spark Program Execution time on `temperature-big csv` is `4 minutes`



App ID	App Name	Attempt ID	Started	Completed	Duration	Spark User	Last Updated
application_1523278295385_2817	Lab1 Q1 include stattion number		2018/05/07 11:19:16	2018/05/07 11:23:14	4.0 min	x_rabsh	2018/05/07 11:23:14

Spark 1.6.0 — Jobs | Stages | Storage | Environment | Executors — Lab1 Q1 include stattion number application UI

Spark Jobs (?)

Total Uptime: 4.0 min
Scheduling Mode: FIFO
Completed Jobs: 3

Event Timeline

Completed Jobs (3)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	saveAsTextFile at NativeMethodAccessorImpl.java:-2	2018/05/07 11:23:07	7 s	2/2 (1 skipped)	129/129 (128 skipped)
1	sortBy at q1-a.py:29	2018/05/07 11:23:07	0.4 s	1/1 (1 skipped)	128/128 (128 skipped)
0	sortBy at q1-a.py:29	2018/05/07 11:19:28	3.6 min	2/2	256/256

Non Parallel python Program Execution

`3028.12795806 second which is equal to 50 minutes`

Reason

It is because spark excute a program in parallel on distributed environment so its excution time is less than non parallel excution of program.

Quesrion 2

Part A

```
#import spark libraries
from pyspark import SparkContext
from operator import add
#spark context object
sc = SparkContext(appName = "Lab1 Q2-count-records")
#read temperature
temperature_file = sc.textFile("/user/x_rabsh/data/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))
#filter data year 1950 t0 2014 and temperature greater than 10
temperature = lines.filter(lambda x: int(x[1][0:4])>=1950 and int(x[1][0:4])
                           <= 2014 and float(x[3]) > 10)
year_temperature = temperature.map(lambda x: (x[1][0:7], 1))
count = year_temperature.reduceByKey(add)
#repartion data and save
monthly_temperatures_count = count.repartition(1)
monthly_temperatures_count.saveAsTextFile("lab1_q2a")
```

part-00000 (q2)

```
(u'1976-04', 3153)
(u'1965-04', 4512)
(u'2000-09', 63837)
(u'2012-11', 255)
(u'1986-07', 55741)
(u'1958-08', 25613)
(u'1975-01', 22)
(u'1989-11', 1126)
(u'1956-06', 21075)
(u'1993-09', 19915)
(u'1998-04', 9479)
(u'1961-08', 38200)
(u'1955-01', 2)
(u'1967-08', 54468)
(u'2013-10', 37839)
(u'1964-09', 25975)
(u'2007-06', 103046)
(u'2011-05', 59106)
(u'2008-07', 126973)
(u'1969-06', 45711)
(u'2013-03', 33)
(u'1971-03', 26)
(u'1975-07', 64408)
(u'1977-09', 28097)
(u'1999-09', 89418)
(u'1988-03', 14)
(u'1972-11', 993)
(u'1962-10', 13842)
(u'1998-02', 169)
(u'1958-11', 21)
(u'1994-06', 43100)
(u'2011-12', 425)
(u'1960-05', 14333)
(u'1970-04', 323)
(u'2009-08', 128349)
(u'1991-01', 2)
(u'2010-11', 414)
(u'1961-11', 395)
(u'1962-03', 1)
(u'1974-08', 64470)
```

Part B

```
from pyspark import SparkContext
#count distinct elements
def count_distinct(a,b):
    return (a[0], a[1]+b[1])

sc = SparkContext(appName = "Lab1 Q2-distnict elements")
#read a data
temperature_file = sc.textFile("/user/x_rabsh/data/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))
#filter temperature 1950 to 2014 and temperature greater than 10
temperature = lines.filter(lambda x: int(x[1][0:4])>=1950 and int(x[1][0:4])
                           <= 2014 and float(x[3]) > 10)
year_temperature = temperature.map(lambda x: (x[1][0:7], (x[0],1)
)).distinct()
count = year_temperature.reduceByKey(lambda v1, v2: count_distinct(v1,v2))
#count elements
count = count.map(lambda x: (x[0], x[1][1]))
#repartition data and save
monthly_temperatures_count = count.repartition(1)
monthly_temperatures_count.saveAsTextFile("lab1_q2b")
```

part-00000(q2-b)

```
(u'1954-01', 3)
(u'1998-09', 326)
(u'1961-05', 268)
(u'1950-12', 1)
(u'1981-07', 329)
(u'1982-11', 180)
(u'1982-06', 339)
(u'2010-05', 319)
(u'1962-08', 301)
(u'1979-09', 351)
(u'1970-06', 369)
(u'1953-09', 117)
(u'1991-03', 162)
(u'2003-08', 320)
(u'1997-09', 340)
(u'1989-06', 315)
(u'1973-10', 349)
(u'1956-02', 2)
(u'1963-06', 310)
(u'2001-11', 150)
(u'1960-10', 103)
(u'1986-03', 14)
(u'1973-07', 370)
(u'1956-11', 4)
(u'1954-08', 119)
(u'1960-07', 126)
(u'1990-08', 316)
(u'1959-07', 126)
(u'2001-02', 10)
(u'1994-04', 269)
(u'1988-01', 2)
(u'1950-04', 36)
(u'1955-03', 1)
(u'2013-12', 8)
(u'1987-08', 320)
(u'1980-09', 338)
(u'2011-07', 319)
(u'2007-04', 263)
(u'2008-05', 316)
(u'1992-11', 39)
(u'2002-03', 154)
```

Quesrion 3

```
# importing spark context
from pyspark import SparkContext
# function for calculating max temperature
def max_temp(a,b):
    if a >= b:
        return a
    else:
        return b

# function for calculating min temperature
def min_temp(a,b):
    if a <= b:
        return a
    else:
        return b
# defining spar context
sc = SparkContext(appName = "lab1-q3")
# reading temperature data file
temperature_file = sc.textFile("/user/x_samza/data/temperature-readings.csv")
# splitting columns in data
lines = temperature_file.map(lambda line: line.split(";"))
# generating key-value pair with year-month-date and station as key and
temperature as value
year_temperature = lines.map(lambda x: ((x[1][0:10],x[0]),float(x[3])))
# filtering data between 1950 and 2014
year_temperature = year_temperature.filter(lambda x: int(x[0][0][0:4]) >=
1950 and int(x[0][0][0:4]) <= 2014)
# Calculating max temperature
max_temperatures = year_temperature.reduceByKey(max_temp)
# Calculating min temperature
min_temperature = year_temperature.reduceByKey(min_temp)
# Joining max_temperature RDD and min temperature RDD
max_min_temperatures = max_temperatures.join(min_temperature)
# Adding min and max temperature and associating count value to use later
monthly_temp = max_min_temperatures.map(lambda x: ((x[0][0][0:7],x[0][1]),
(float(x[1][0])+float(x[1][1]),2) ))
# Adding previous and current temperature and count on the basis of key
monthly_avg_temp = monthly_temp.reduceByKey(lambda (temp1, count1), (temp2,
count2): (temp1+temp2, count1+count2))
# mapping required columns and calculating avg by dividing the sum of daily
avg by totat no. of days in month
monthly_avg_temp = monthly_avg_temp.map(lambda ((date, station), (sumTemp,
sumCount)): ((date, station), sumTemp/float(sumCount)))
# repartitioning to form single RDD
max_temperature_monthly= monthly_avg_temp.repartition(1)
# Sorting result in descending order
max_temperature_monthly = max_temperature_monthly.sortBy(ascending = False ,
keyfunc = (lambda k : k[1]))
```

```
# Saving File
max_temperature_monthly.saveAsTextFile("avg_temperature_station")
```

```
((u'2014-07', u'96000'), 26.3)
((u'1994-07', u'96550'), 23.071052631578947)
((u'1983-08', u'54550'), 23.0)
((u'1994-07', u'78140'), 22.970967741935485)
((u'1994-07', u'85280'), 22.872580645161293)
((u'1994-07', u'75120'), 22.858064516129037)
((u'1994-07', u'65450'), 22.85645161290322)
((u'1994-07', u'96000'), 22.80806451612903)
((u'1994-07', u'95160'), 22.764516129032256)
((u'1994-07', u'86200'), 22.711290322580652)
((u'2002-08', u'78140'), 22.700000000000006)
((u'1994-07', u'76000'), 22.698387096774194)
((u'1997-08', u'78140'), 22.666129032258063)
((u'1994-07', u'105260'), 22.659677419354832)
((u'1975-08', u'54550'), 22.642857142857142)
((u'2006-07', u'76530'), 22.598387096774196)
((u'1994-07', u'86330'), 22.548387096774192)
((u'2006-07', u'75120'), 22.52741935483871)
((u'1994-07', u'54300'), 22.469354838709677)
((u'2006-07', u'78140'), 22.45806451612903)
((u'2001-07', u'96550'), 22.40833333333333)
((u'2010-07', u'98180'), 22.379032258064516)
((u'2006-07', u'65450'), 22.377419354838707)
((u'1994-07', u'85210'), 22.375806451612906)
((u'1994-07', u'98180'), 22.36774193548387)
((u'2014-07', u'98180'), 22.36774193548387)
((u'2002-08', u'98180'), 22.366129032258065)
((u'1994-07', u'92100'), 22.31774193548387)
((u'1994-07', u'86470'), 22.30806451612903)
((u'1994-07', u'83230'), 22.27258064516129)
((u'1994-07', u'64290'), 22.259677419354833)
((u'1994-07', u'97490'), 22.258064516129036)
((u'1994-07', u'94180'), 22.25322580645161)
((u'1972-07', u'173960'), 22.244999999999997)
((u'1994-07', u'74080'), 22.241935483870968)
((u'2006-07', u'54300'), 22.23709677419355)
((u'2002-08', u'98210'), 22.2354838709677 37)
((u'1994-07', u'106070'), 22.23225806451613)
((u'1994-07', u'75100'), 22.229032258064517)
((u'1994-07', u'53440'), 22.1975)
((u'1994-07', u'83270'), 22.177419354838705)
((u'1994-07', u'103080'), 22.164516129032258)
```

Quesrion 4

```
# importing Spark Context
from pyspark import SparkContext
# Function to calculate maximum temperature and precipitation
def max_fun(a,b):
    if a >= b:
        return a
    else:
        return b
# Function to calculate sum of daily precipitation
def daily_prec(a,b):
    return a+b
# adding spark context
sc = SparkContext(appName = "lab1-q4")
# reading temperature data
temperature_file = sc.textFile("/user/x_samza/data/temperature-readings.csv")
# reading precipitation data
precipitation_file = sc.textFile("/user/x_samza/data/precipitation-
readings.csv")
# splitting temperature data into columns
lines = temperature_file.map(lambda line: line.split(";"))
# forming a key value pair keeping station as key and temperature as value
station_temperature = lines.map(lambda x: (x[0],float(x[3])))
# finding max temperature for each station
max_temperatures = station_temperature.reduceByKey(max_fun)
# filtering max temperature between 25 and 30 degrees
max_temperatures = max_temperatures.filter(lambda x: x[1] >= 25 and x[1] <=
30)
# splitting precipitation data into columns
lines_prec = lines = precipitation_file.map(lambda line: line.split(";"))
# forming a key value pair keeping date and station as key and temperature as
value
station_precipitation = lines_prec.map(lambda x: ((x[0],x[1]),float(x[3])))
# calculating sum of precipitation for each day for each station
daily_precipitation = station_precipitation.reduceByKey(daily_prec)
# creating a key-value pair keeping station as key and daily precipitation as
value
daily_precipitation = daily_precipitation.map(lambda x:
(x[0][0],float(x[1])))
# calculating max daily precipitation for each station
max_precipitation_station = daily_precipitation.reduceByKey(max_fun)
# filtering max daily precipitation between 100 and 200 mm
max_precipitation_station = max_precipitation_station.filter(lambda x: x[1]
>= 100 and x[1] <= 200)
# joining result of max temperature and max daily precipitation on the basis
of stations
max_result = max_temperatures.join(max_precipitation_station)
# saving result
max_result.saveAsTextFile("prec_temp_station")
```

Because there is no matching between 2 Rdd, so result will be null.

Quesrion 5

```
# importing required context
from pyspark import SparkContext
from operator import add

sc = SparkContext(appName = "Lab1 Q5")
#reading data
ostergotland_file = sc.textFile("/user/x_samza/data/stations-
Ostergotland.csv")
#partition data
stations = ostergotland_file.map(lambda line: line.split(";"))
map_oster = stations.map(lambda x: (x[0]) )
# collect and broadcast Stations
stations = map_oster.distinct().collect()
Os_stations = sc.broadcast(stations)
precipitation_file = sc.textFile("/user/x_samza/data/precipitation-
readings.csv")
precipt_lines = precipitation_file.map(lambda line: line.split(";"))
#filter stations
filter_precipt = precipt_lines.filter(lambda x: x[0] in Os_stations.value )
#making key-value pair keeping year-month and stations as key and
precipitation as value
map_precipt =  filter_precipt.map(lambda x: ((x[1][0:7],x[0]),float(x[3])))
#calculating total monthly precipitation
monthly_prec = map_precipt.reduceByKey(add)
#find average and count
avg_prec = monthly_prec.map(lambda x: ((x[0][0], (float(x[1]) ,1)  )))
result = avg_prec.reduceByKey(lambda x,y: ( (x[0] + y[0]) , (x[1] + y[1]) ))
result = result.map(lambda x: (x[0] , float(x[1][0] / x[1][1])))
#partition and save data
output = result.repartition(1)
output.saveAsTextFile("lab1_q5")
```

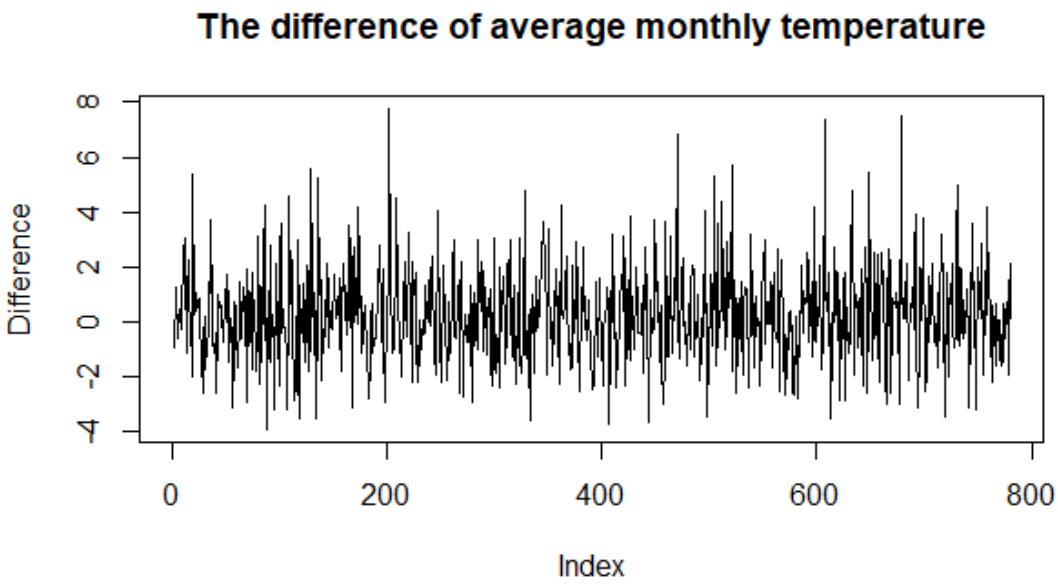
```
(u'1996-11', 67.11666666666665)
(u'2008-03', 42.200000000000024)
(u'2008-10', 59.56666666666669)
(u'2014-05', 58.000000000000014)
(u'2001-11', 26.38333333333334)
(u'2011-05', 37.85)
(u'1999-07', 29.083333333333343)
(u'2010-02', 52.75000000000005)
(u'2013-08', 54.07500000000001)
(u'2002-06', 98.7833333333333)
(u'2013-05', 47.92500000000001)
(u'1998-11', 28.966666666666683)
(u'2002-03', 26.933333333333337)
(u'2013-02', 25.525000000000016)
(u'2007-08', 54.166666666666664)
(u'1993-07', 95.39999999999999)
(u'2000-06', 62.01666666666667)
(u'2009-02', 24.78333333333335)
(u'1996-08', 37.716666666666676)
(u'2007-02', 33.066666666666684)
(u'1993-11', 42.80000000000003)
(u'2009-07', 113.16666666666664)
(u'2014-12', 35.46250000000001)
(u'2005-01', 18.05)
(u'2004-12', 24.25)
(u'2005-04', 11.65)
(u'2009-11', 64.21666666666665)
(u'1994-05', 25.100000000000005)
(u'2009-08', 61.566666666666684)
(u'1996-02', 15.733333333333334)
(u'2015-05', 93.22499999999998)
(u'2003-01', 17.716666666666672)
(u'1996-05', 63.23333333333335)
(u'2000-01', 18.616666666666674)
(u'2001-07', 40.283333333333346)
(u'1995-12', 5.116666666666666)
(u'2001-02', 36.766666666666694)
(u'2011-12', 42.133333333333375)
(u'2002-12', 20.916666666666668)
```

Quesrion 6

```
from pyspark import SparkContext
from operator import add
sc = SparkContext(appName = "lab1 q6")
staOstergotland_file = sc.textFile("/user/x_samza/data/stations-
Ostergotland.csv")
lines = staOstergotland_file.map(lambda line: line.split(";"))
#collect and broadcast
stations = lines.map(lambda x: int(x[0]))
stations = stations.distinct().collect() #collect to a python list
```

```
Os_stations = sc.broadcast(stations)
#read temperature file
temperature_file = sc.textFile("/user/x_samza/data/temperature-readings.csv")
lines_tempFile = temperature_file.map(lambda line: line.split(";"))
#Filter out Ostergoland stations
temperatures = lines_tempFile.filter(lambda x: (int(x[1][0:4]) >= 1950 and
int(x[1][0:4]) <= 2014 and (x[0] in Os_stations.value)))
#monthly average temperature
map_temperature = temperatures.map(lambda x: ((x[1], int(x[0])),
(float(x[3]), float(x[3]))))
min_max_temp = map_temperature.reduceByKey(lambda x, y:(min(x[0], y[0]),
max(x[1], y[1])))
min_max_temp = min_max_temp.map(lambda x: ((x[0][0][0:7], x[0][1]),
(x[1][0]+x[1][1], 2)))
min_max_temp = min_max_temp.reduceByKey(lambda x,y: (x[0]+y[0], x[1]+y[1]))
average_by_month = min_max_temp.map(lambda x: ( (x[0][0]), (float(x[1][0]/
x[1][1]) , 1)))
#year average
#average_by_year = average_by_month.map(lambda x: (x[0], (x[1], 1)))
average_by_year = average_by_month.reduceByKey(lambda x, y: (x[0] + y[0],
x[1] + y[1]))
average_by_year = average_by_year.map(lambda x: (x[0], x[1][0] / x[1][1] ))
#filter year 1950 to 1980
average_longTerm = average_by_year.filter(lambda x: (int(x[0][0:4]) >= 1950
and int(x[0][0:4]) <= 1980))
average_longTerm = average_longTerm.map(lambda x: (x[0][5:7], (x[1], 1)))
average_longTerm = average_longTerm.reduceByKey(lambda x, y: (x[0] + y[0],
x[1] + y[1]))
average_longTerm=average_longTerm.map(lambda x: (x[0], x[1][0]/x[1][1] ))
#collecting the result and map with average year
results = average_longTerm.collect()
monthAvg = {month: temperature for (month, temperature) in results}
difference_temp = average_by_year.map(lambda x: (x[0], abs(x[1])-
abs(monthAvg.get(x[0][5:7], 0))))
#save result to output
difference_results = difference_temp.repartition(1)\
            .saveAsTextFile("lab1_q6")
```

```
(u'1997-04', -0.921165815367428)
(u'1978-03', -0.04657464630409491)
(u'2003-05', 0.27568038621108215)
(u'2004-04', 1.2260719468703334)
(u'1981-10', -0.6171810863434155)
(u'1981-03', 0.42682936903540414)
(u'1977-06', 0.24203697467407004)
(u'2009-07', -0.2640071126699617)
(u'2003-12', -0.08261718510417759)
(u'1986-11', 2.7764981707166143)
(u'1968-02', 0.4523795721334647)
(u'1952-03', 3.0368000435222084)
(u'1950-09', 0.34317939855842994)
(u'1974-07', -1.1745224038304158)
(u'1990-05', 2.241061617882636)
(u'1955-11', 0.45937300602623177)
(u'1982-02', 0.28350362049889277)
(u'1952-10', -2.0341972153756744)
(u'2010-01', 5.360916378106804)
(u'1996-03', 0.2819613338447885)
(u'2005-03', 1.8684715977744066)
(u'2006-02', 0.39860102309629486)
(u'1978-10', 0.501429335492813)
(u'1980-04', 0.6892537650521522)
(u'1954-05', 0.8847712953019915)
(u'1967-07', 0.6908193780436278)
(u'2002-11', -1.9892330545798287)
(u'1964-06', -0.18498225609515906)
(u'1951-02', -2.6065613145660427)
(u'2007-09', -0.9795695191904876)
(u'2005-10', 0.21786290192637825)
(u'2008-08', -1.280330356428168)
(u'1987-12', 0.07361797758259814)
(u'1972-01', 1.4377717807686583)
(u'1969-09', 1.0229229883020192)
(u'1975-08', 3.724891840729512)
(u'1958-01', 0.5291861728282119)
(u'1961-01', -0.6325756137722842)
(u'1987-05', -1.1370762120293882)
```


The difference of average monthly temperature
Difference
-4
-2
0
2
4
6
8
0
200
400
600
800
Index