DAT470/DIT065 Assignment 3

Xinyan Liu
liuxinya@chalmers.se

Leah Wanja Ndirangu
leahw@chalmers.se

2025-05-11

Problem 1

(a) Implement the missing bits of pyspark twitter follows.py to determine the maximum number of people followed, the Twitter id of the account with the maximum number of people followed, the average number of people followed, and the number of accounts that follow no-one

Below is the implementation of the `map` function in Python:

```
#!/usr/bin/env python3

import time
import argparse
import findspark
findspark.init()
from pyspark import SparkContext

def mapper(line):
    user_follow = line.split(':')
    user = user_follow[0].strip()
    follow = user_follow[1].strip().split()
    return (user, len(follow))

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description = \
                                    'Compute Twitter follows
                                        .')
    parser.add_argument('-w','--num-workers',default=1,type=
        int,
                            help = 'Number of workers')
    parser.add_argument('filename',type=str,help='Input
        filename')
    args = parser.parse_args()

    start = time.time()
    sc = SparkContext(master = f'local[{args.num_workers}]')

    lines = sc.textFile(args.filename)
```

```python
    header = lines.first()
    data = lines.map(mapper).reduceByKey(lambda x, y: x+y)
    no_of_user = data.count()
    total_followed = data.values().sum()
    average = total_followed/no_of_user
    max_follow = data.max(key= lambda x: x[1])[0]
    max_follow_times = data.max(key= lambda x: x[1])[1]
    follow_no_one = data.filter(lambda line: line[1] == 0).
        count()

    end = time.time()

    total_time = end - start

    # the first ??? should be the twitter id
    print(f'max follows: {max_follow} follows {
        max_follow_times}')
    print(f'users follow on average: {average}')
    print(f'number of user who follow no-one: {follow_no_one
        }')
    print(f'num workers: {args.num_workers}')
    print(f'total time: {total_time}')
```

The graph 1 and table 1 below shows our result when running it on 10M dataset as required.

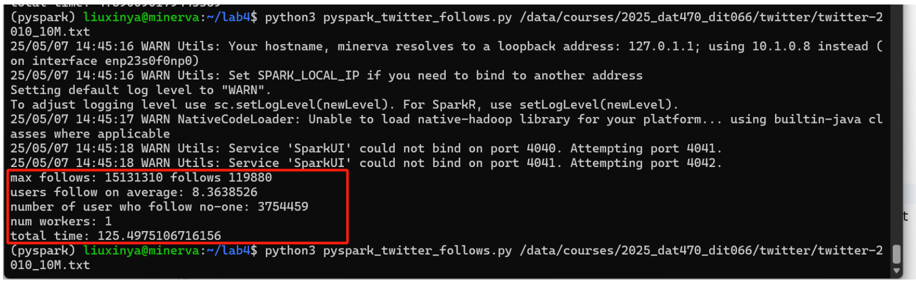


Figure 1: Results for program

Table 1: Twitter Follow Statistics

Metric	**Value**
Max follows	15,131,310
Follows per user on average	8.3638526
Number of users who follow no one	3,754,459
Number of workers	1
Total time (seconds)	125.4975

(b) Measure the scalability of your algorithm on 1, 2, 4, . . . , 64 cores. Plot the empirical speedup as the function of cores. In addition to the plot, report the single-core runtime on the dataset

The following graph 2 shows the empirical speedup as the function of cores $S_n = \frac{t_1}{t_n}$.

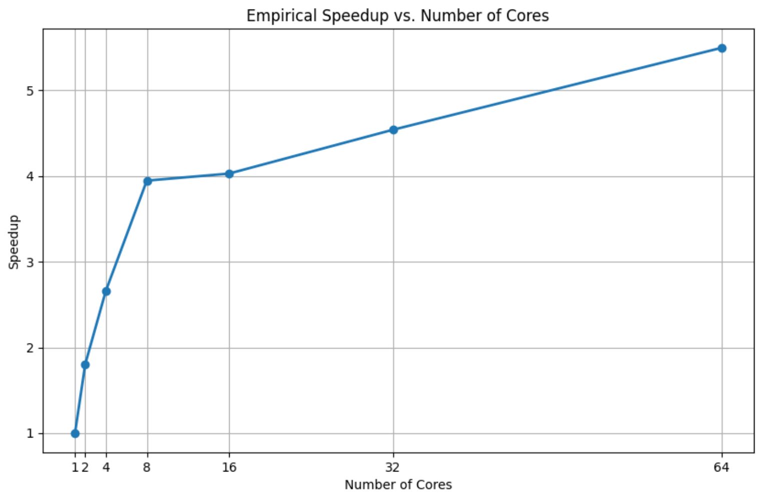


Figure 2: Empirical speedup as the function of cores

The following is the table 2 for running time.

Table 2: Execution Time for Different Core Counts

No. of Worker	**Execution Time (s)**
1	71.5186
2	39.6128
4	26.9117
8	18.1231
16	17.7543
32	15.7526
64	13.0140

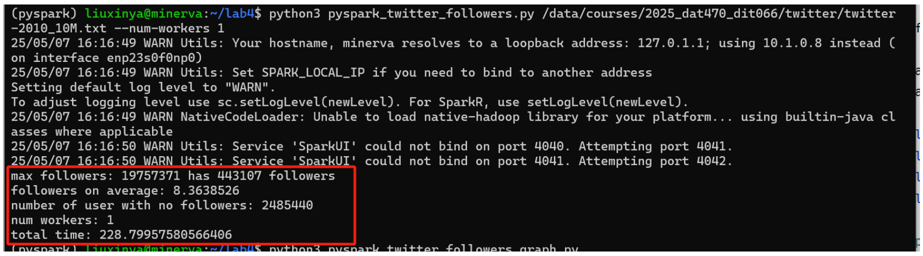
The running time for single core is 13.0140s.

(c) Implement the missing bits of pyspark twitter followers.py to deter- mine the maximum number of followers, the Twitter id of the account with the maximum number of followers, the average number of followers, and the number of accounts that have no followers.

```
#!/usr/bin/env python3

import time
import argparse
import findspark
findspark.init()
from pyspark import SparkContext


def mapper(line):
    user_follow = line.split(':')
    user = user_follow[0].strip()
    follows = user_follow[1].strip().split()
    return [(user, 0)] + [(follow,1) for follow in follows]


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description = \
                                    'Compute Twitter
                                        followers.')
    parser.add_argument('-w','--num-workers',default=1,type=
        int,
                            help = 'Number of workers')
    parser.add_argument('filename',type=str,help='Input
        filename')
    args = parser.parse_args()

    start = time.time()
    sc = SparkContext(master = f'local[{args.num_workers}]')

    lines = sc.textFile(args.filename)

    data = lines.flatMap(mapper).reduceByKey(lambda x,y:x+y)

    total_no_user = data.count()
    total_no_follower = data.values().sum()

    average = total_no_follower / total_no_user

    most_follower_id = data.max(key= lambda x: x[1])[0]
    most_follower_times = data.max(key= lambda x: x[1])[1]

    no_follower = data.filter(lambda x:x[1] == 0).count()

    end = time.time()

    total_time = end - start
```

```python
    # the first ??? should be the twitter id
    print(f'max followers: {most_follower_id} has {
        most_follower_times} followers')
    print(f'followers on average: {average}')
    print(f'number of user with no followers: {no_follower}'
        )
    print(f'num workers: {args.num_workers}')
    print(f'total time: {total_time}')
```

The graph 3 and table 3 below shows our result when running it on 10M dataset as required.



```
(pyspark) liuxinya@minerva:~/lab4$ python3 pyspark_twitter_followers.py /data/courses/2025_dat470_dit066/twitter/twitter
-2010_10M.txt --num-workers 1
25/05/07 16:16:49 WARN Utils: Your hostname, minerva resolves to a loopback address: 127.0.1.1; using 10.1.0.8 instead (
on interface enp23s0f0np0)
25/05/07 16:16:49 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/05/07 16:16:49 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
25/05/07 16:16:50 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
25/05/07 16:16:50 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
max followers: 19757371 has 443107 followers
followers on average: 8.3638526
number of user with no followers: 2485440
num workers: 1
total time: 228.79957580566406
```

Figure 3: Results for program

Table 3: Twitter Follower Statistics

Metric	**Value**
Max followers	19,757,371
Followers of max follower user	443,107
Followers per user on average	8.3638526
Number of users with no followers	2,485,440
Number of workers	1
Total time (seconds)	228.79957580566406

(d) Measure the scalability of your algorithm on 1, 2, 4, . . . , 64 cores. Plot the empirical speedup as the function of cores. In addition to the plot, report the single-core runtime on the dataset

The following graph 4 shows the empirical speedup as the function of cores $S_n = \frac{t_1}{t_n}$.

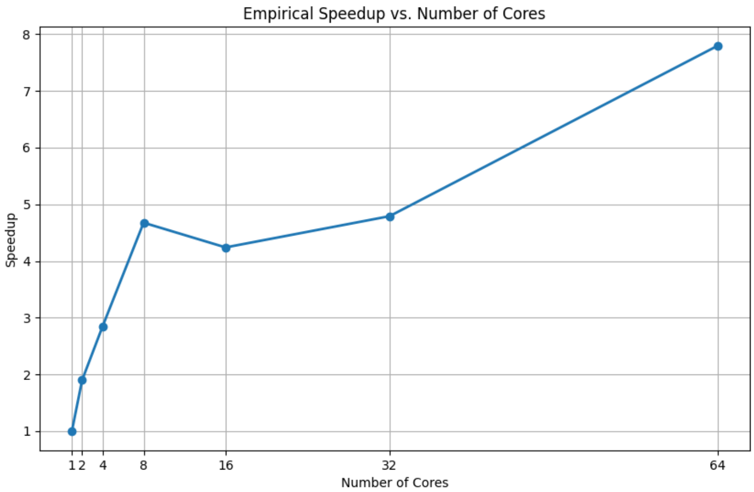


Figure 4: Empirical speedup as a function of cores

The following is the table 4 for running time.

Table 4: Execution Time for Different Core Counts

Core Count	**Execution Time (s)**
1	228.7996
2	120.3873
4	80.2875
8	48.9231
16	53.9507
32	47.7505
64	29.3489

The running time for single core is 29.3489s.

Problem 2

(a) Using pyspark climate.py as your starting point, implement a Spark pro- gram that, using pyspark.sql Data Frames, computes the values requested, that is,

Our code can refer to the pyspark_climate.py/assigment4_problem2.py (too long to display here), and the following are some of our results on the tiny, small, and medium datasets

Table 5: station code, station name, and the slope of the top 5 stations tiny

station code	station name, and the slope of the to
A007016932 at STE CATHERINE	QC CA BETA=row1.341e-03 °F/d
USC00333345 at GREEN	OH US BETA=row1.053e-03 °F/d
USW00094732 at NORTHEAST PHILADELPHIA AIRPORT	PA US BETA=row8.829e-04 °F/d
JA000047770 at KOBE	JA BETA=row7.927e-04 °F/d
NOE00109939 at TRYVASSHOGDA	NO BETA=row7.032e-04 °F/d

Fraction of positive coefficients 0.7105263157894737

Table 6: Five-number summary of BETA values

parameter	values
beta$_{m}in$	-6.942e-03
beta$_{q}1$	-2.033e-05
beta$_{m}edian$	1.514e-04
beta$_{q}3$	6.698e-04
beta$_{m}ax$	7.559e-03

Table 7: station code, station name, and the slope of the top 5 stations tiny

station code	station name, and the Top 5 differences temperature
USC00313976 at HENDERSONVILLE 1 NE	NC US difference 5.7 °C
USC00398472 at TYNDALL	SD US difference 3.6 °C
USC00410493 at BALLINGER 2 NW	TX US difference 1.6 °C
USC00401790 at CLARKSVILLE WWTP	TN US difference 0.7 °C
USC00144712 at LINCOLN 1 SE	KS US difference 0.5 °C

Fraction of positive differences: 0.8333333333333334
Five-number summary of decade average difference values:

Table 8: station code, station name, and the slope of the top 5 stations small

station code	Five-number summary of decade average difference values: small
$tdiff\_min$	-2.7 °C
$tdiff\_q1$	0.5 °C
$tdiff\_median$	0.7 °C
$tdiff\_q3$	3.6 °C
$tdiff\_max$	5.7 °C

Table 9: station code, station name, and the slope of the top 5 stations Medium

station code	station name, and the slope of the top 5 stations Med
MXN00005023 at PALESTINA DGE	MX BETA=row7.559e-03 °F/d
MXN00009052 at UNIDAD MODELO	MX BETA=row5.161e-03 °F/d
MXN00009003 at AQUILES SERDAN 46	MX BETA=row2.004e-03 °F/d
SWE00139268 at FREDRIKSBERG	SW BETA=row1.421e-03 °F/d
CA003052995 at HAILSTONE BUTTE LO	AB CA BETA=row1.133e-03 °F/d

Fraction of positive coefficients 0.9

Table 10: Five-number summary of BETA values:

operator	value
$beta\_min$	$-1.012e-03$
$beta\_q1$	$1.651e-04$
$beta\_median$	$6.102e-04$
$beta\_q3$	$8.829e-04$
$beta\_max$	$1.341e-03$

======medium====

Table 11: station code, station name, and the slope of the top 5 stations tiny

station code	Name Temperature difference ($\circ$ C)
USC00213455 at HALLOCK	MN US difference 16.1 °C
USC00264349 at LAHONTAN DAM	NV US difference 15.7 °C
RSE00151755 at KON KOLODEZ	RS difference 13.9 °C
CA007051200 at CAUSAPSCAL	QC CA difference 13.7 °C
USC00256290 at O NEILL	NE US difference 13.4 °C

Fraction of positive differences: 0.7794117647058824
Five-number summary of decade average difference values:

Table 12: Five-number summary of decade average difference values:

parameter	values
$tdiff\_min$	-23.6 °C
$tdiff\_q1$	0.3 °C
$tdiff\_median$	5.3 °C
$tdiff\_q3$	8.6 °C
$tdiff\_max$	16.1 °C

num workers: 1 total time: 95.8 s

(b) determine the scalability of your solution. Use the large dataset for your measurements. Plot the speedup as the function of CPU cores for w = 1, 2, 4, . . . , 64 workers

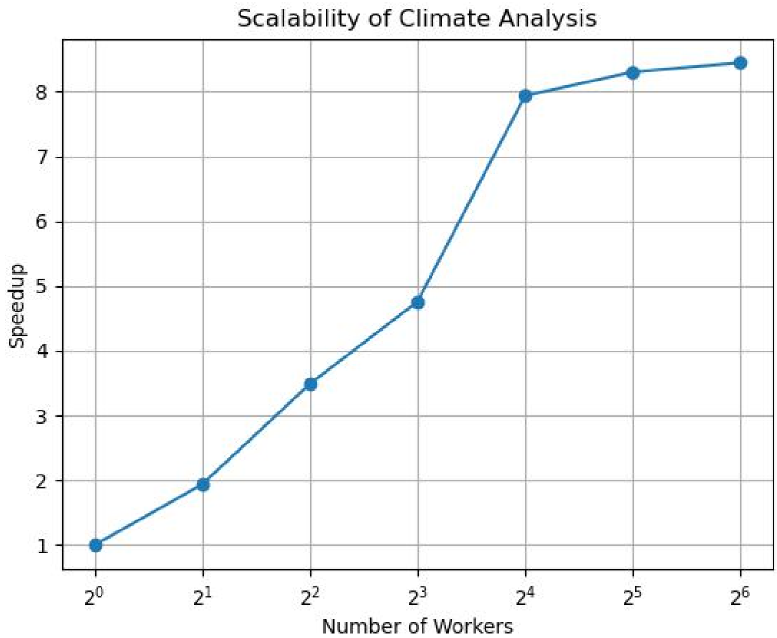


Figure 5: Results for program

(c) Determine a breakdown of the time: how large fraction of the time is spent in reading the data, and how much in computations. What is the speedup you get for computations only using 64 cores

The two screenshots 6 7 below demonstrate our running results for both 1 cores (worker) and 64 cores (workers) on the full dataset.

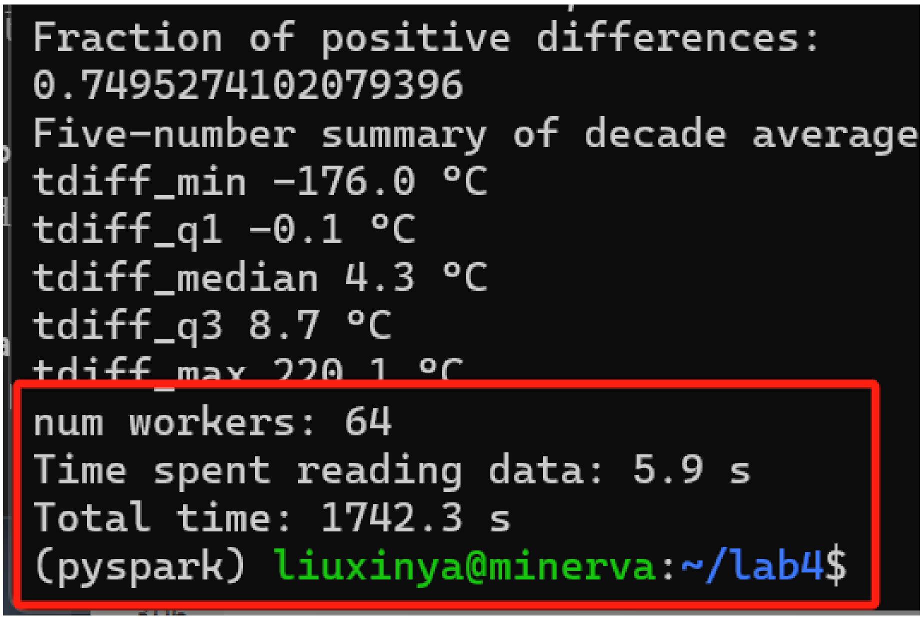


Figure 6: Results with 64 workers

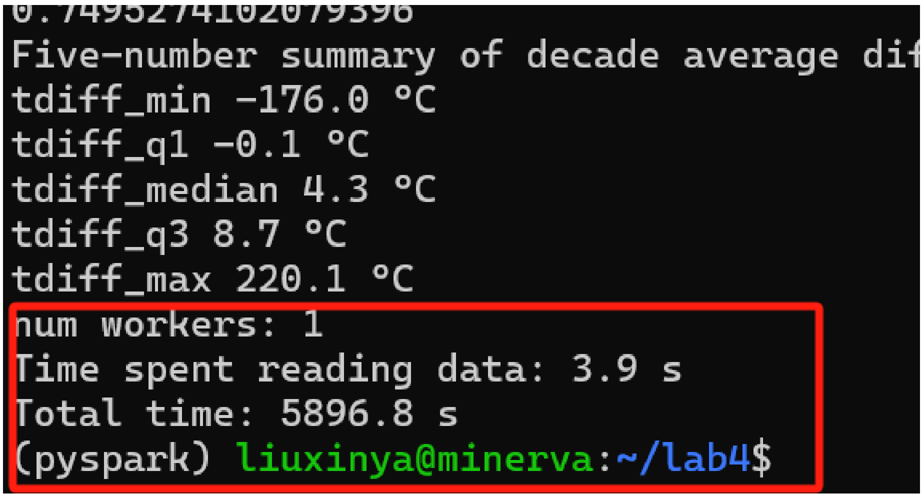


Figure 7: Results for 1 worker

The table below shows the time for "reading data" and "computation" with 1 worker and 64 workers.

Table 13: Comparison of Time Spent with Different Number of Workers

Workers	Time Spent Reading Data (s)	Total Time (s)	Time for Computation (s)
1	3.9	5896.8	5892.9
64	5.9	1742.3	1736.4

and the speed up for computation for 64 cores is 5892.9/1736.4 = 3.3937

(d) Compute the aforementioned values for the full dataset. Record the station codes, names, and values for the top 5 slopes and temperature differences as nicely formatted tables in your report. Also report the five-number summaries as nicely formatted tables. Record the fraction of positive values out of all values computed. Also record the total wall-clock running time of your code and how many workers you used

1. Top 5 Slopes/Coefficients 14

Table 14: Top 5 Coefficients (Slopes)

Station Code	Station Name	BETA (°F/d)
USC00364611	KITTANNING LOCK 7, PA US	1.085e+02
USC00241993	COOKE, MT US	8.400e+01
USC00211585	CLEARWATER, MN US	4.400e+01
USC00114363	INA, IL US	3.450e+01
USC00205667	MOUNT PLEASANT, MI US	3.300e+01

2. Top 5 Temperature Differences 15

Table 15: Top 5 Temperature Differences

Station Code	Station Name	Difference (°C)
RSM00024944	OLEKMINSK, RS	220.1
RSM00024641	VILJUJSK, RS	210.9
RSM00024266	VERHOJANSK, RS	200.5
RSM00021921	KJUSJUR, RS	186.8
KZ000035078	ATBASAR, KZ	103.1

3. Five-Number Summary of BETA Values 16

Table 16: Five-Number Summary of BETA Values

Statistic	Value (°F/d)
Minimum	-3.035e+02
First Quartile (Q1)	-2.997e-04
Median	4.314e-04
Third Quartile (Q3)	1.279e-03
Maximum	1.085e+02

4. Fraction of Positive Differences 19

Table 17: Fraction of Positive Differences

Metric	Value
Fraction of Positive Differences	0.7495

5. Five-Number Summary of Decade Average Difference Values 18

Table 18: Five-Number Summary of Temperature Differences

Statistic	Value (°C)
Minimum	-176.0
First Quartile (Q1)	-0.1
Median	4.3
Third Quartile (Q3)	8.7
Maximum	220.1

6. Fraction of Positive Differences 19

Table 19: Fraction of Positive Differences

Metric	Value
Fraction of Positive Differences	0.7495

7. Total Running Time and Workers 20

Table 20: Running Time and Workers

Metric	Value
Number of Workers	1
Time Spent Reading Data	3.9 s
Total Time	5896.8 s

(e) The data has not been sanitized and some of the measurements (particulary very old ones) are likely erroneous, which may lead to some non-sensical values. Still, the five-number summaries should give us a hint at the big picture. What are your thoughts: what does the data tell us about climate change

The dataset gives a clear overview of rising temperatures across globe. It shows there are large atterns of global warming.