Big Data Analysis Project 2:Weather

Aishwarya Patil

M12981344.
Computer Science Engineering.
University of Cincinnati.
Cincinnati, Ohio
patilay@mail.uc.edu

TABLE OF CONTENTS

INTRODUCTION

This project report solves all the assigned tasks, on the give weather data from https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/, they are:

1. Average TMIN, TMAX for each year excluding abnormalities or missing data.

2. Maximum TMAX, Minimum TMIN for each year excluding abnormalities or missing data.

3. 5 hottest, 5 coldest weather stations for each year excluding abnormalities or missing data.

4. Hottest and coldest day and corresponding weather stations in the entire dataset.

Bonus tasks:

1. Median TMIN, TMAX for each year and corresponding weather stations.

2. Median TMIN, TMAX for the entire dataset

The first 4 tasks are completed using **MapReduce**.

Step 1

Steps in execution for Map Reduce code.
*Write mapper.py and reducer.py.
*Then change access permission of mapper and reducer.
*Run the Mapreduce on Hadoop

```
   $ hadoop jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar -
file /home/patilay/mapper.py -mapper /home/patilay/mapper.py -file
/home/patilay/reducer.py -reducer /home/patilay/reducer.py -input
/user/tatavag/weather  -output /tmp/weatherOutput/
```

*Get the result

```
   $ hdfs dfs -cat /tmp/weatherOutput/part*
```

Mapper.py

```
#!/usr/bin/env python

import sys
import csv


f = csv.reader(sys.stdin)

for line in f:

    missingAttr = (line[3] == -9999 or line[4] == 'P')

    tminTmaxAttr = (line[2] == 'TMIN' or line[2] == 'TMAX')

    qualityReading = (line[5] == '')

    sourceCheck = (line[6] != '')

    if (not missingAttr and tminTmaxAttr and qualityReading and sourceCheck):

        print '%s\t%s\t%s\t%s' % (line[1], line[0], line[2], line[3])
```

Reducer.py

```
#!/usr/bin/env python

from operator import itemgetter
import sys

sum_tmax = 0
sum_tmin = 0

min_tmin = 0
max_tmax = 0

cnt_tmax = 0
cnt_tmin = 0

hottest_data = { 'value': 0, 'stations': [] }
coldest_data = { 'value': 0, 'stations': [] }

cur_year = None
year = None

for line in sys.stdin:
    line = line.strip()
    timestamp, station_id, element, value = line.split('\t', 3)
    year = str(timestamp)[:4]
```

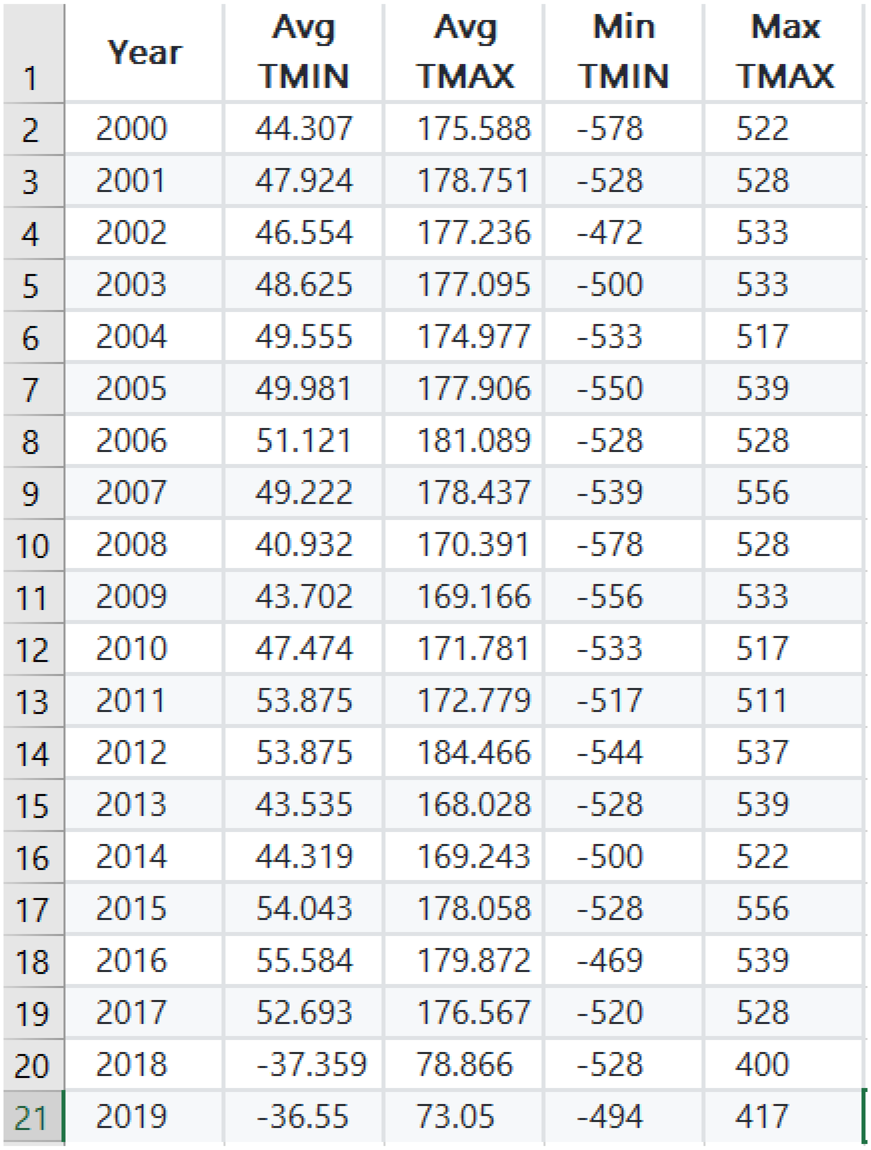
```
try:
    value = int(value)
except ValueError:
    continue

if cur_year is None:
    cur_year = year

if year != cur_year:
    print 'Year: %s' % cur_year
    print 'Average TMAX\t%s\t' % (sum_tmax * 1.0 / cnt_tmax)
    print 'Average TMIN\t%s\t' % (sum_tmin * 1.0 / cnt_tmin)
    print 'Max TMAX\t%s\t' % max_tmax
    print 'Min TMIN: %s' % min_tmin
    print 'Coldest Stations %s' % ([x['station_id'] for x in coldest_data['stations']])
    print 'Coldest Station Values %s' % ([x['value'] for x in coldest_data['stations']])
    print 'Hottest Stations %s' % ([x['station_id'] for x in hottest_data['stations']])
    print 'Hottest Station Values %s' % ([x['value'] for x in hottest_data['stations']])
    print '--------------------------------'

    cur_year = year
    sum_tmax = 0
    sum_tmin = 0
    cnt_tmax = 0
    cnt_tmin = 0
    max_tmax = 0
    min_tmin = 0
    hottest_data['stations'] = []
    coldest_data['stations'] = []

if element == "TMAX":
    sum_tmax += value
    if value > max_tmax:
        max_tmax = value
    cnt_tmax += 1

elif element == "TMIN":
    sum_tmin += value
    if value < min_tmin:
        min_tmin = value
    cnt_tmin += 1

if value > hottest_data['value']:
    hottest_data = {
            'value': value,
            'station_id': station_id,
            'day': timestamp,
            'stations': hottest_data['stations']
        }

    if value < coldest_data['value']:
        coldest_data = {
            'value': value,
            'station_id': station_id,
            'day': timestamp,
            'stations': coldest_data['stations']
        }

    if len(hottest_data['stations']) < 5:
        hottest_data['stations'].append({ 'value': value, 'station_id': station_id })
    else:
        coolest_value = min([x['value'] for x in hottest_data['stations']])
        if value > coolest_value:
            idx = next((index for (index, d) in enumerate(hottest_data['stations']) if d['value'] ==
coolest_value), None)
            hottest_data['stations'][idx] = {
                'value': value,
                'station_id': station_id
            }

    if len(coldest_data['stations']) < 5:
        coldest_data['stations'].append({ 'value': value, 'station_id': station_id })
    else:
        hottest_value = max([x['value'] for x in coldest_data['stations']])
        if value < hottest_value:
            idx = next((index for (index, d) in enumerate(coldest_data['stations']) if d['value'] ==
hottest_value), None)
            coldest_data['stations'][idx] = {
                'value': value,
                'station_id': station_id
            }

if year == cur_year:
    print 'Year: %s' % cur_year
    print 'Average TMAX: %s\t' % (sum_tmax * 1.0 / cnt_tmax)
    print 'Average TMIN: %s\t' % (sum_tmin * 1.0 / cnt_tmin)
    print 'Max TMAX: %s' % max_tmax
    print 'Min TMIN: %s' % min_tmin
    print 'Hottest Day:\t%s\t%s\t%s' %(hottest_data['day'], hottest_data['value'],
hottest_data['station_id'])
    print 'Coldest Day:\t%s\t%s\t%s' %(coldest_data['day'], coldest_data['value'],
coldest_data['station_id'])

    print 'Coldest Stations %s' % ([x['station_id'] for x in coldest_data['stations']])
    print 'Coldest Station Values %s' % ([x['value'] for x in coldest_data['stations']])
    print 'Hottest Stations %s' % ([x['station_id'] for x in hottest_data['stations']])
    print 'Hottest Station Values %s' % ([x['value'] for x in hottest_data['stations']])
```

TASK 1 & 2

This code gives data for following analysis:

- Average TMIN, TMAX for each year excluding abnormalities or missing data

- Maximum TMAX, Minimum TMIN for each year excluding abnormalities or missing data



1	Year	Avg TMIN	Avg TMAX	Min TMIN	Max TMAX
2	2000	44.307	175.588	-578	522
3	2001	47.924	178.751	-528	528
4	2002	46.554	177.236	-472	533
5	2003	48.625	177.095	-500	533
6	2004	49.555	174.977	-533	517
7	2005	49.981	177.906	-550	539
8	2006	51.121	181.089	-528	528
9	2007	49.222	178.437	-539	556
10	2008	40.932	170.391	-578	528
11	2009	43.702	169.166	-556	533
12	2010	47.474	171.781	-533	517
13	2011	53.875	172.779	-517	511
14	2012	53.875	184.466	-544	537
15	2013	43.535	168.028	-528	539
16	2014	44.319	169.243	-500	522
17	2015	54.043	178.058	-528	556
18	2016	55.584	179.872	-469	539
19	2017	52.693	176.567	-520	528
20	2018	-37.359	78.866	-528	400
21	2019	-36.55	73.05	-494	417

TASK 3

5 hottest , 5 coldest weather stations for each year excluding abnormalities or missing data

	Year	Stations
Hottest Stations	2000	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319
Coldest Stations		USC00505644 USC00501684 USC00508140 USC00501684 USC00505644

Hottest Stations	2001	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319
Coldest Stations		USS0051R01S USR0000ABCA USW00026508 USW00026508 USW00026508
Hottest Stations	2002	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319
Coldest Stations		USR0000AKAI USC00503212 USS0051R01S USR0000ABEV USS0050S01S
Hottest Stations	2003	USC00042319 USR0000AHAV USR0000CMEA USC00042319 USC00042319
Coldest Stations		USW00026533 USC00501492 USS0051R01S USS0051R01S USC00501492
Hottest Stations	2004	USC00024761 USC00024761 USC00042319 USC00042319 USC00042319
Coldest Stations		USC00501684 USC00501684 USC00502350 USC00502568 USW00026412
Hottest Stations	2005	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319
Coldest Stations		USC00509313 USC00501684 USC00501684 USC00501684 USC00501684
Hottest Stations	2006	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319

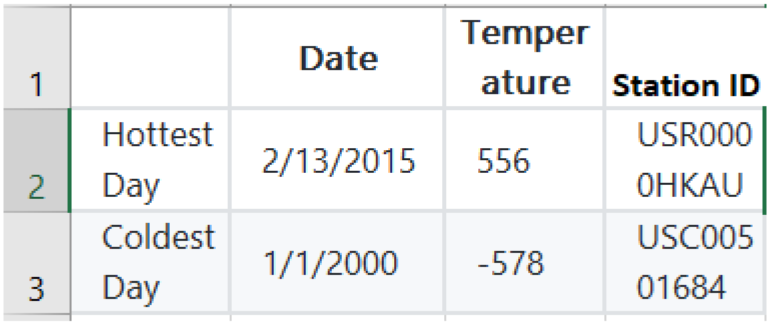
Coldest Stations		USC00501492 USC00501492 USR0000ASEL USC00501492 USR0000ABEV
Hottest Stations	2007	USR0000CSAW USC00042319 USR0000CSAW USC00042319 USW00053139
Coldest Stations		USS0045R01S USC00501684 USC00501684 USC00501684 USC00501684
Hottest Stations	2008	USC00044297 USC00042319 USC00024761 USC00042319 USC00044297
Coldest Stations		USC00501684 USC00501684 USC00501684 USC00501684 USC00501684
Hottest Stations	2009	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319
Coldest Stations		USC00502101 USC00502101 USC00501684 USC00502101 USC00501684
Hottest Stations	2010	USC00042319 USC00042319 USC00042319 USR0000AHAV USC00042319
Coldest Stations		USC00501684 USC00502101 USS0051R01S USC00502101 USC00501684
Hottest Stations	2011	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319
Coldest Stations		USC00509869 USS0045R01S USS0051R01S USS0045R01S USS0051R01S
Hottest Stations	2012	USC00042319 USS0005N23S USC00042319 USC00042319 USC00042319

Coldest Stations		USC00503212 USC00503165 USC00503165 USS0051R01S USC00503165
Hottest Stations	2013	USC00042319 USW00004134 USC00042319 USC00042319 USW00004134
Coldest Stations		USC00502339 USC00501684 USC00502339 USC00501684 USC00501684
Hottest Stations	2014	USC00042319 USW00053139 USC00042319 USC00042319 USC00042319
Coldest Stations		USC00501684 USC00501684 USC00501684 USC00501684 USC00501684
Hottest Stations	2015	USR0000HKAU USC00042319 USC00042319 USR0000HKAU USR0000HKAU
Coldest Stations		USC00501684 USC00501684 USC00502339 USC00502339 USC00502339
Hottest Stations	2016	USR0000CBEV USC00042319 USC00040924 USC00042319 USC00042319
Coldest Stations		USR0000ACHL USR0000ACHL USR0000ACHL USC00501684 USS0051R01S
Hottest Stations	2017	USC00042319 USC00042319 USC00042319 USC00021050 USC00042319
Coldest Stations		USW00026529 USS0051R01S USS0051R01S USW00026529 USR0000ASLC
Hottest Stations	2018	USC00042319 USC00042319 USC00042319 USC00042319 USC00042319
Coldest Stations		USC00501684 USR0000ANOR USR0000AKAV USC00501684 USC00096406

Hottest Stations	2019	USW00012907 USC00417624 USW00022010 USC00415048 USR0000TFAL
Coldest Stations		USC00211840 USC00501684 USC00509891 USC00211840 USC00218618

TASK 4

Hottest and coldest day and corresponding weather stations in the entire dataset



		Date	Temperature	Station ID
1		Date	Temperature	Station ID
2	Hottest Day	2/13/2015	556	USR0000HKAU
3	Coldest Day	1/1/2000	-578	USC00501684

BONUS TASKS:

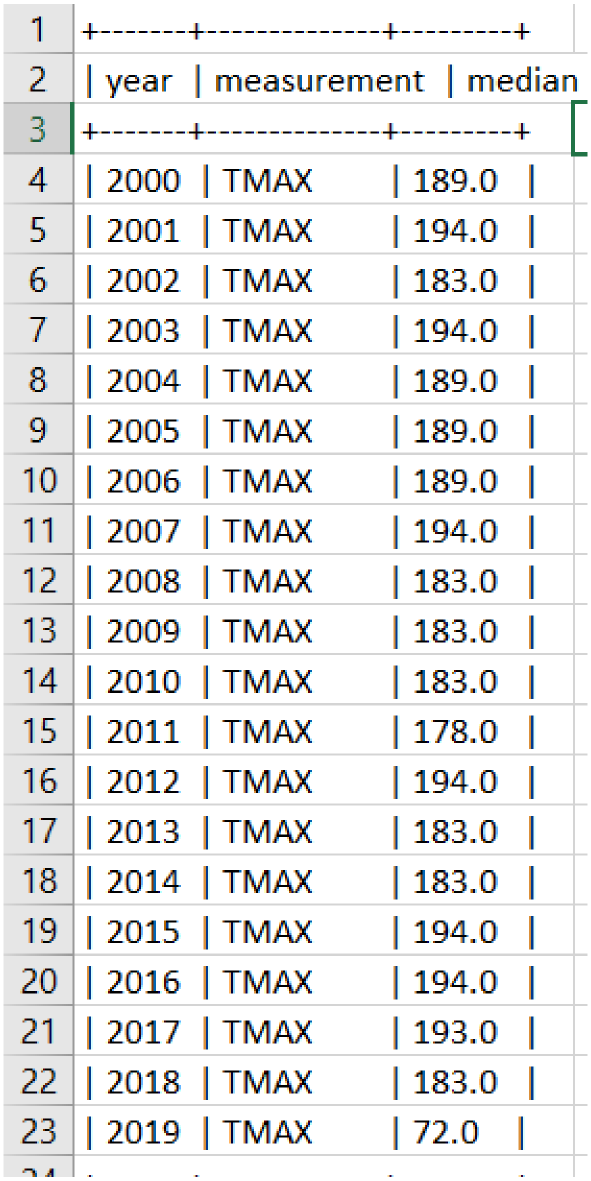
Used Hive for calculating median. HiveQL Query will result in Median TMIN and TMAX for each year. Create a external table first.

```
CREATE EXTERNAL TABLE IF NOT EXISTS weather_table ( station STRING,  date_id INT,
measurement STRING, VALUE INT, flag1 STRING, flag2 STRING, flag3 STRING, flag4 STRING)
COMMENT 'Data about weather' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED
AS TEXTFILE location '/user/patilay/weather';
```

1.Median TMIN, TMAX for each year and corresponding weather stations

```
SELECT year, measurement, percentile(cast(value AS BIGINT),0.5) Median FROM

(

 SELECT *, substr(cast(date_id as STRING) ,1,4) year FROM

 weather_table

 WHERE measurement='TMAX' AND flag2=''

) z1

GROUP BY year,measurement
```

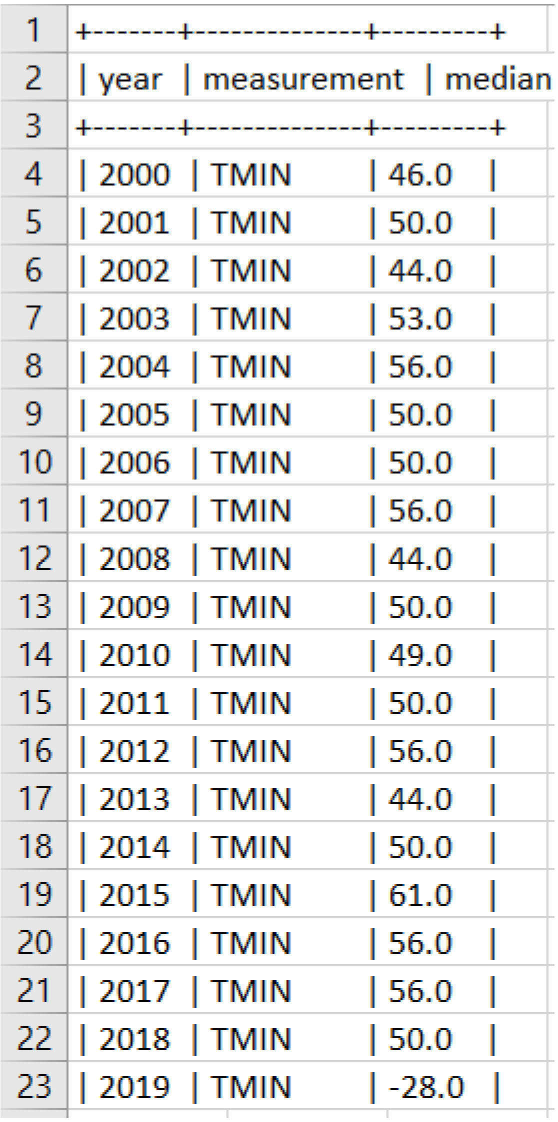
ORDER BY year;



```
+--------+--------------+---------+
| year   | measurement  | median
+--------+--------------+---------+
| 2000  | TMAX         | 189.0   |
| 2001  | TMAX         | 194.0   |
| 2002  | TMAX         | 183.0   |
| 2003  | TMAX         | 194.0   |
| 2004  | TMAX         | 189.0   |
| 2005  | TMAX         | 189.0   |
| 2006  | TMAX         | 189.0   |
| 2007  | TMAX         | 194.0   |
| 2008  | TMAX         | 183.0   |
| 2009  | TMAX         | 183.0   |
| 2010  | TMAX         | 183.0   |
| 2011  | TMAX         | 178.0   |
| 2012  | TMAX         | 194.0   |
| 2013  | TMAX         | 183.0   |
| 2014  | TMAX         | 183.0   |
| 2015  | TMAX         | 194.0   |
| 2016  | TMAX         | 194.0   |
| 2017  | TMAX         | 193.0   |
| 2018  | TMAX         | 183.0   |
| 2019  | TMAX         | 72.0    |
```

```
SELECT year, measurement, percentile(cast(value AS BIGINT),0.5) Median FROM

(

 SELECT *, substr(cast(date_id as STRING) ,1,4) year FROM

 weather_table

 WHERE measurement='TMIN' AND flag2=''

) z2
```

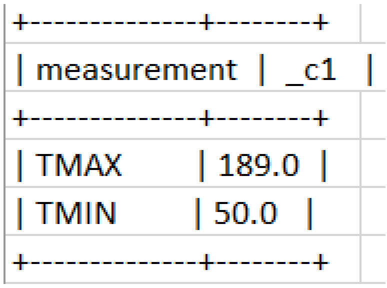
```
GROUP BY year,measurement

ORDER BY year;
```



```
+-------+--------------+---------+
| year  | measurement  | median
+-------+--------------+---------+
| 2000  | TMIN         | 46.0    |
| 2001  | TMIN         | 50.0    |
| 2002  | TMIN         | 44.0    |
| 2003  | TMIN         | 53.0    |
| 2004  | TMIN         | 56.0    |
| 2005  | TMIN         | 50.0    |
| 2006  | TMIN         | 50.0    |
| 2007  | TMIN         | 56.0    |
| 2008  | TMIN         | 44.0    |
| 2009  | TMIN         | 50.0    |
| 2010  | TMIN         | 49.0    |
| 2011  | TMIN         | 50.0    |
| 2012  | TMIN         | 56.0    |
| 2013  | TMIN         | 44.0    |
| 2014  | TMIN         | 50.0    |
| 2015  | TMIN         | 61.0    |
| 2016  | TMIN         | 56.0    |
| 2017  | TMIN         | 56.0    |
| 2018  | TMIN         | 50.0    |
| 2019  | TMIN         | -28.0   |
```

2.Median TMIN, TMAX for the entire dataset

```
SELECT measurement, percentile(cast(value as BIGINT),0.5)

FROM weather_table

WHERE flag2='' and flag3!=''

GROUP BY measurement

ORDER BY measurement;
```



```
+--------------+--------+
| measurement  |  _c1   |
+--------------+--------+
| TMAX         | 189.0  |
| TMIN         | 50.0   |
+--------------+--------+
```

Why did I use Map Reduce for the first 4 tasks? Writing mappers and reducers was easier because of the previous homework assignment. But it wasn't possible rather too difficult to calculate median in map reduce that's why switched to Hive. But the execution took lot more time for map reduce code.

Why did I use Hive QL to solve Bonus Tasks instead of MapReduce?

Firstly, the SQL like interface made it much easier to write queries specific to the task. Furthermore, it optimizes the query in the background which eventually get converted in to map-reduce jobs. However, the speed is by far better than other tools like pySpark. I have tried to solve the Bonus Tasks using pySpark initially , but the computation took almost 20 min whereas the simple hive query gave results in 2 min. Hence, the computation speed is a boon to HiveQL.

References

1. http://rare-chiller-615.appspot.com/mr1.html

2. http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/