NAME-ISTIAQUE MANNAFEE SHAIKAT

ID-303527

Followed instructions the below link to install hadoop-2.8.0

https://github.com/MuhammadBilalYar/Hadoop-On-Window/wiki/Step-by-step-Hadoop-2.8.0-installation-on-Window-10

Exercise 1.2: Basic hadoop operations

1.

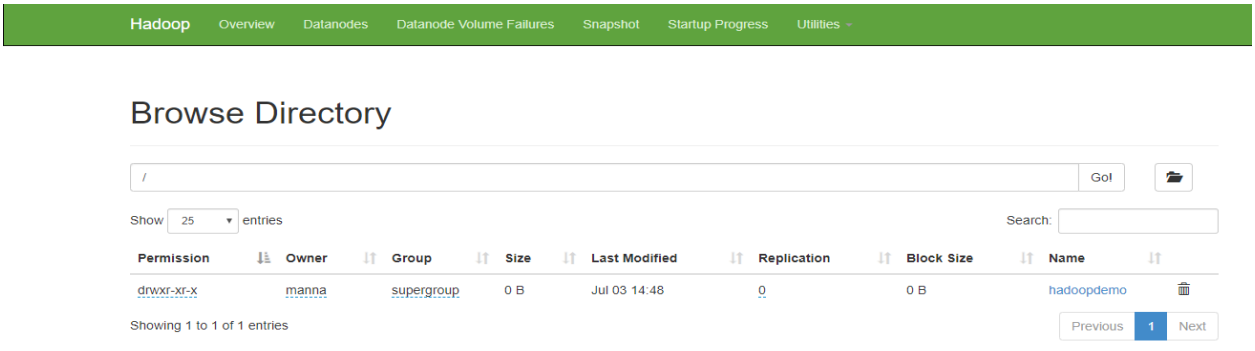
```
C:\hadoop-2.8.0\sbin>hadoop version
Hadoop 2.8.0
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 91f2b7a13d1e97be65db92ddabc627cc29ac0009
Compiled by jdu on 2017-03-17T04:12Z
Compiled with protoc 2.5.0
From source with checksum 60125541c2b3e266cbf3becc5bda666
This command was run using /C:/hadoop-2.8.0/share/hadoop/common/hadoop-common-2.8.0.jar
```

2.

```
C:\hadoop-2.8.0\sbin>hadoop fs -ls /
Found 1 items
drwxr-xr-x   - manna supergroup          0 2019-07-03 14:48 /hadoopdemo
```

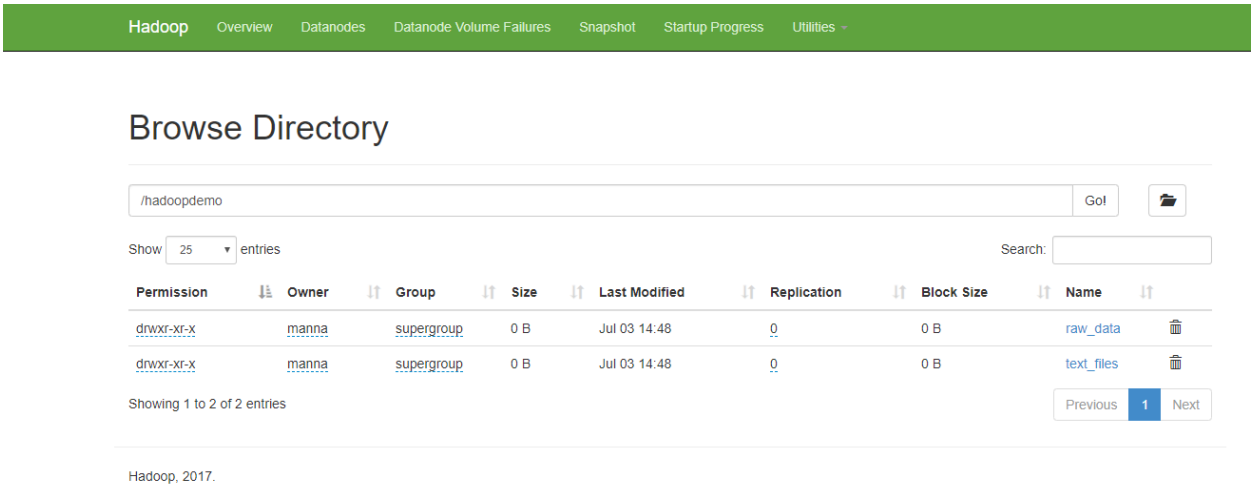
3.

```
C:\hadoop-2.8.0\sbin>hadoop fs -mkdir /hadoopdemo
```



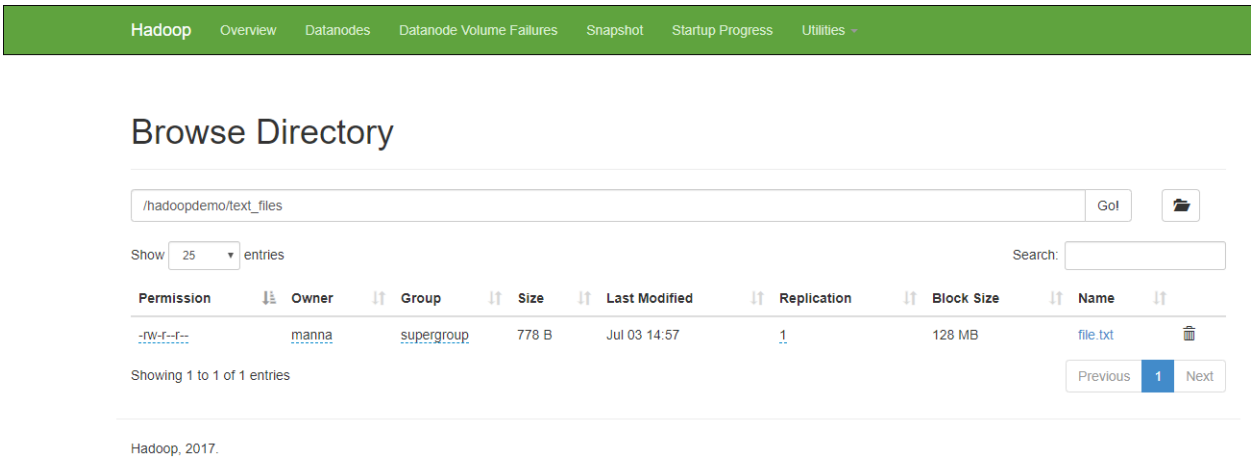
4.

```
C:\hadoop-2.8.0\sbin> hadoop fs -mkdir /hadoopdemo/text_files

C:\hadoop-2.8.0\sbin> hadoop fs -mkdir /hadoopdemo/raw_data
```



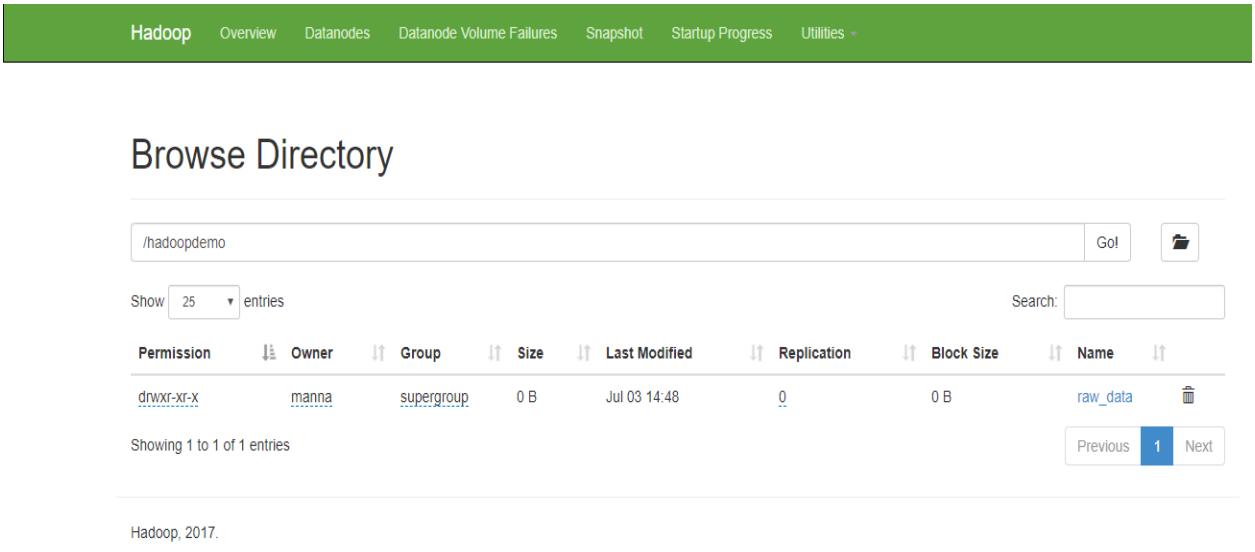
5.

```
C:\hadoop-2.8.0\sbin>hadoop fs -put C:\Users\manna\desktop\file.txt /hadoopdemo/text_files
```



7.

```
C:\hadoop-2.8.0\sbin>hadoop fs -rm -r /hadoopdemo/text_files
Deleted /hadoopdemo/text_files
```



8 & 9.

BEFORE

```
C:\hadoop-2.8.0\sbin>hadoop fs -cat /hadoopdemo/text_files/file.txt
https://www.google.com/maps/dir/Bromberger+Str.+44,+31141+Hildesheim,+Germany/PSV+Green+White+Hildesheim+e.V.,+Marienburger+Stra%C3%9Fe,+Hildesheim/@52.1340656,9.9692939,16z/am=t/data=!4
bb817!2m2!1d9.9737509!2d52.138727!1m5!1m1!1s0x47baae3f6062ee71:0x65d9561fd127fad0!2m2!1d9.9785292!2d52.129663!2m3!6e0!7e2!8j1561021320!3e3
9,50 euro

https://www.google.com/maps/dir/Bromberger+Str.+44,+31141+Hildesheim,+Germany/DJK+Blau-Wei%C3%9F+Hildesheim,+Lucienv%C3%B6rder+Allee+8,+31139+Hildesheim/@52.1434274,9.9471582,15z/data=!4
bb817!2m2!1d9.9737509!2d52.138727!1m5!1m1!1s0x47baafa1b275a3c5:0x4be7444a9d97324!2m2!1d9.9413713!2d52.1415661!2m3!6e0!7e2!8j1561021320!3e3
8,00 euro
```

AFTER

```
C:\hadoop-2.8.0\sbin>hadoop fs -cat /hadoopdemo/text_files/file.txt
https://www.google.com/maps/dir/Bromberger+Str.+44,+31141+Hildesheim,+Germany/PSV+Green+White+Hildesheim+e.V.,+Marienburger+Stra%C3%9Fe,+Hildesheim/@52.1340656,9.9692939,16z/am=t/data=!4m18!4m17!1m5!1m1!1s0x47baae45a9343ccd:0x19a11
bb817!2m2!1d9.9737509!2d52.138727!1m5!1m1!1s0x47baae3f6062ee71:0x65d9561fd127fad0!2m2!1d9.9785292!2d52.129663!2m3!6e0!7e2!8j1561021320!3e3
9,50 euro

https://www.google.com/maps/dir/Bromberger+Str.+44,+31141+Hildesheim,+Germany/DJK+Blau-Wei%C3%9F+Hildesheim,+Lucienv%C3%B6rder+Allee+8,+31139+Hildesheim/@52.1434274,9.9471582,15z/data=!4m18!4m17!1m5!1m1!1s0x47baae45a9343ccd:0x19a11
bb817!2m2!1d9.9737509!2d52.138727!1m5!1m1!1s0x47baafa1b275a3c5:0x4be7444a9d97324!2m2!1d9.9413713!2d52.1415661!2m3!6e0!7e2!8j1561021320!3e3
8,00 euro

hello world!!
C:\hadoop-2.8.0\sbin>
```

Exercise 1.3: Word Count Map Reduce example

1. Created directory to store the text file in "input" directory after word count get the output in "output folder".

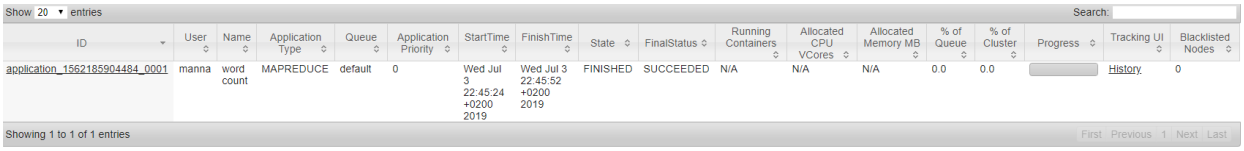
```
C:\hadoop-2.8.0>hadoop fs -mkdir /input

C:\hadoop-2.8.0>hadoop fs -mkdir /output
```

2. Transfer and store the text file "big.txt" from local system to hadoop system.

```
C:\hadoop-2.8.0>hadoop fs -put C:\Users\manna\Desktop\big.txt /input
```

3. Run the **word count** MapReduce job provided in

%HADOOP_HOME%>bin\yarn jar share\hadoop\mapreduce\hadoop-mapreduce-examples-2.8.0.jar /input output



Show 20 entries — Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1562185904484_0001	manna	word count	MAPREDUCE	default	0	Wed Jul 3 22:45:24 +0200 2019	Wed Jul 3 22:45:52 +0200 2019	FINISHED	SUCCEEDED	N/A	N/A	N/A	0.0	0.0		History	0

Showing 1 to 1 of 1 entries — First Previous 1 Next Last

First few output are given

```
stories!        1
stories!"       1
stories)        1
stories,        6
stories,"       1
stories.        2
storing 2
storm   23
storm,  2
storm-tossed    1
storm.  5
storm;  1
storm?" 1
stormcloud      2
stormed 8
stormed!        1
storming        1
storms  3
storms. 1
stormy  7
stormy, 1
story   83
story!  2
story), 1
story,  13
story--they     1
story-teller.   1
story.  21
story." 2
story:  1
story?" 1
stout   44
stout,  15
stout-built,    1
stouter 3
stouter!"       1
stouter!..."    1
stouter?"       1
stoutest        2
stoutly 3
stoutness,      2
stove   3
stove.  3
```

Exercise 2: Analysis of Airport efficiency with Map Reduce (10 points)

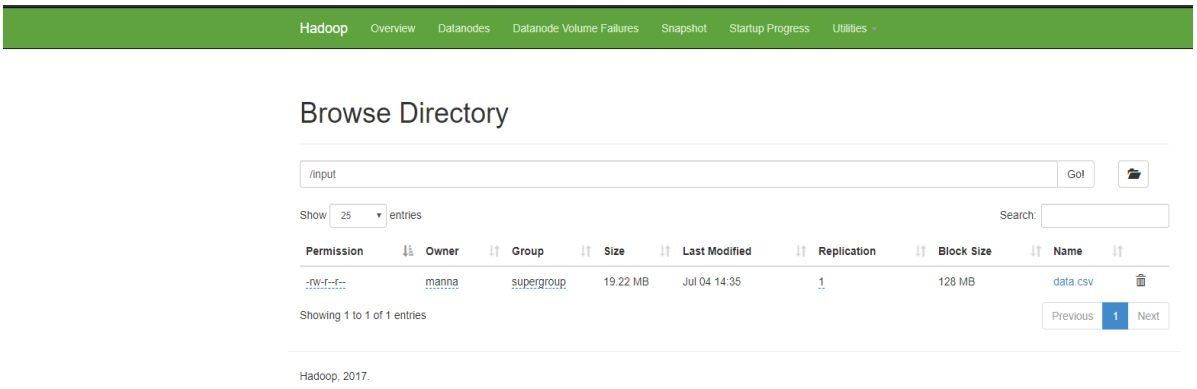
The following are the steps which are being followed while doing map reduce on Airport efficiency.

1. Two directories were created in hdfs i.e input(to store input data) and output(to get output data)

```
C:\hadoop-2.8.0\sbin>hadoop fs -mkdir /input

C:\hadoop-2.8.0\sbin>hadoop fs -mkdir /output
```

2. Uploaded the .csv file in hdfs input folder

```
C:\hadoop-2.8.0\bin>hadoop fs -put C:/Users/manna/desktop/data.csv /input
```



3. Write Mapper and Reducer in python

In Hadoop, there is a Java program called Hadoop streaming-jar. This program internally read (stdin) and print out (stdout) line by line. Therefore, Python can read each line as a string and parse it by using functions like strip and split (","). For example, the first line would be parsed like this.

[2017-01-01, 20409, 264, SEA, JFK, 2102, 2.00, 0506, 1.00] -> [‘2017-01-01’,’20409’,’264’,’SEA’,’JFK’,’2102’,’2.00’,’0506’,’1.00’]

So, need to only pick the fourth element (airport) and the ninth element (arrival delay)in each array, and print out (stdout) them as Key-Value pair.

Here is Mapper.py code

```
#!/usr/bin/python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    line = line.strip()
    line = line.split(",")

    if len(line) >=8:
        airport = line[3]
        arrdelay = line[8]

        print ('%s\t%s' % (airport, arrdelay))
```

In reducer, the string is parsed which is coming from mapper.py as Key-Value pair, and keep them in dictionary like {airport:[arrival delay,.....]}

```
#!/usr/bin/python

#Reducer.py
import sys


(last_airport,max_arrdelay)=(None,0)
(last_airport,min_arrdelay)=(None,0)
airport_arrdelay = {}
ave_delay_rank={}



for line in sys.stdin:
    data = line.strip().split('\t')

    if len(data) != 2:
        # Something has gone wrong. Skip this line.
        continue
    airport,arrdelay = data

    if airport in airport_arrdelay:
        airport_arrdelay[airport].append(int(arrdelay))
    else:
        airport_arrdelay[airport] = []
        airport_arrdelay[airport].append(int(arrdelay))

    if last_airport and last_airport!=airport:
        print('maximum arrival delay:','%s\t%s'% (last_airport, max_arrdelay))
        print('minimum arrival delay:','%s\t%s'% (last_airport, min_arrdelay))

        (last_airport,max_arrdelay)=(airport,int(arrdelay))
        (last_airport,min_arrdelay)=(airport,int(arrdelay))
    else:
        (last_airport,max_arrdelay)=(airport,max(max_arrdelay,int(arrdelay)))
        (last_airport,min_arrdelay)=(airport,min(min_arrdelay,int(arrdelay)))


for airport in airport_arrdelay.keys():
    ave_delay = sum(airport_arrdelay[airport])*1.0 / len(airport_arrdelay[airport])
    ave_delay_rank[airport] = []
    ave_delay_rank[airport].append(float(ave_delay))
    print ('average arrival delay','%s\t%s'% (airport, ave_delay))
sorted_x = sorted(ave_delay_rank.items(), key=lambda kv: kv[1])
print("Top 10 airports by their average Arrival delay")
count=0
for keys,values in sorted_x:
    if count<10:
        print('%s\t%s'% (keys,values[0]))
        count+=1

```

3.1 Explaining the reducer code

Collecting data line by line by strip and split function as well as performed a quick check if I have data of length less than 2.If the data is correct then the 'data' is unpacked and saved in 'airport' and 'arrdelay'

```
for line in sys.stdin:
    data = line.strip().split('\t')

    if len(data) != 2:
        # Something has gone wrong. Skip this line.
        continue
    airport,arrdelay = data
```

This if else is performed to create a dict of airport and arrival delay time so that it can be used later on to find the average arrival delay time for each airport.

```
    if airport in airport_arrdelay:
        airport_arrdelay[airport].append(int(arrdelay))
    else:
        airport_arrdelay[airport] = []
        airport_arrdelay[airport].append(int(arrdelay))
```

This where the maximum and minimum is time is found for each airport.First the data is sorted after coming out from the mapper.Then in every iteration the currentkey(airport) is compared with the oldkey(last_airport) to find the minimum and maximum for each airport**.**

```
if last_airport and last_airport!=airport:
    print('maximum arrival delay:','%s\t%s'% (last_airport, max_arrdelay))
    print('minimum arrival delay:','%s\t%s'% (last_airport, min_arrdelay))

    (last_airport,max_arrdelay)=(airport,int(arrdelay))
    (last_airport,min_arrdelay)=(airport,int(arrdelay))
else:
    (last_airport,max_arrdelay)=(airport,max(max_arrdelay,int(arrdelay)))
    (last_airport,min_arrdelay)=(airport,min(min_arrdelay,int(arrdelay)))
```

At the end average arrival delay is calculated for every airport using the airport_arrdelay dictionary which was created earlier.On the other hand, the ave_delay_rank dict holds the key (airport) and value(average arrival delay).Then it is sorted to find the top 10 airports by their average Arrival delay.

```
for airport in airport_arrdelay.keys():
    ave_delay = sum(airport_arrdelay[airport])*1.0 / len(airport_arrdelay[airport])
    ave_delay_rank[airport] = []
    ave_delay_rank[airport].append(float(ave_delay))
    print ('average arrival delay','%s\t%s'% (airport, ave_delay))
sorted_x = sorted(ave_delay_rank.items(), key=lambda kv: kv[1])
print("Top 10 airports by their average Arrival delay")
count=0
for keys,values in sorted_x:
    if count<10:
        print('%s\t%s'% (keys,values[0]))
        count+=1
```

4. Change access permission on Mapper and Reducer

Changed access permission on your mapper and reducer as below.

chmod +x mapper.py
chmod+x reducer.py

5. Tested mapper.py and reducer.py locally

simulated process-flow by just using "cat" command in the terminal

```
mannafee@Mannafee:/mnt/c/Users/manna/desktop$ cat data.csv | python3 mapper.py | sort | python3 reducer.py
```

Showing output of minimum and maximum for the first few airports.

```
mannafee@Mannafee:/mnt/c/Users/manna/desktop$ cat data.csv | python3 mapper.py | sort | python3 reducer.py
maximum arrival delay: ABE      786
minimum arrival delay: ABE      -30
maximum arrival delay: ABI      261
minimum arrival delay: ABI      -20
maximum arrival delay: ABQ      908
minimum arrival delay: ABQ      -47
maximum arrival delay: ABR      1245
minimum arrival delay: ABR      -29
maximum arrival delay: ABY      307
minimum arrival delay: ABY      -42
maximum arrival delay: ACT      192
minimum arrival delay: ACT      -27
maximum arrival delay: ACV      565
minimum arrival delay: ACV      -26
maximum arrival delay: ACY      649
minimum arrival delay: ACY      -43
maximum arrival delay: ADK      41
minimum arrival delay: ADK      -31
maximum arrival delay: ADQ      71
minimum arrival delay: ADQ      -31
maximum arrival delay: AEX      351
```

Showing output of average arrival delay for the first few airports.

```
average arrival delay ABE       15.43010752688172
average arrival delay ABI       28.962962962962962
average arrival delay ABQ       5.056603773584905
average arrival delay ABR       35.03333333333333
average arrival delay ABY       3.654320987654321
average arrival delay ACT       11.142857142857142
average arrival delay ACV       10.952941176470588
average arrival delay ACY       2.139751552795031
average arrival delay ADK       2.3333333333333335
average arrival delay ADQ       -5.225
average arrival delay AEX       9.93061224489796
average arrival delay AGS       16.6470588235294l3
average arrival delay ALB       2.7625994694960214
average arrival delay AMA       0.8148148148148148
average arrival delay ANC       4.973176865046102
average arrival delay APN       20.904761904761905
average arrival delay ASE       32.85406301824212
average arrival delay ATL       9.904612978889757
average arrival delay ATW       25.306532663316585
average arrival delay AUS       6.0779642663779105
average arrival delay AVL       8.51231527093596
average arrival delay AVP       28.821052631578947
average arrival delay AZO       4.115107913669065
average arrival delay BDL       -3.306301050175029
```

Showing output for the top 10 airports by their average Arrival delay

```
Top 10 airports by their average Arrival delay
LSE     -22.3
YAK     -19.3125
PPG     -12.5
CDV     -11.26923076923077
ISN     -9.619047619047619
LBE     -8.7
EAU     -8.018518518518519
ORH     -7.649122807017544
INL     -7.530612244897959
BFL     -6.788571428571428
```

6. Testing mapper and reducer script in Hadoop platform

Using Hadoop streaming-2.8.0 jar to execute the mapper and reducer script
Command "**hadoop jar C:\hadoop-2.8.0\share\hadoop\tools\lib\hadoop-streaming-2.8.0.jar -file mapper.py -mapper "python mapper.py" -file reducer.py -reducer "python reducer.py" -input /input/data.csv -output 14**"

```
C:\hadoop-2.8.0>hadoop jar C:\hadoop-2.8.0\share\hadoop\tools\lib\hadoop-streaming-2.8.0.jar -file mapper.py -mapper "python mapper.py" -file reducer.py -reducer "python reducer.py" -input /input/data.csv -output 14
19/07/05 11:33:31 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper.py, reducer.py] [] C:\Users\manna\AppData\Local\Temp\streamjob2840630007336944374.jar tmpDir=null
19/07/05 11:33:32 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
19/07/05 11:33:32 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
19/07/05 11:33:32 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
19/07/05 11:33:33 INFO mapred.FileInputFormat: Total input files to process : 1
19/07/05 11:33:33 INFO mapreduce.JobSubmitter: number of splits:1
19/07/05 11:33:33 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local822758103_0001
19/07/05 11:33:34 INFO mapred.LocalDistributedCacheManager: Localized file:/C:/hadoop-2.8.0/mapper.py as file:/tmp/hadoop-manna/mapred/local/1562319214005/mapper.py
19/07/05 11:33:34 INFO mapred.LocalDistributedCacheManager: Localized file:/C:/hadoop-2.8.0/reducer.py as file:/tmp/hadoop-manna/mapred/local/1562319214006/reducer.py
19/07/05 11:33:34 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
19/07/05 11:33:34 INFO mapred.LocalJobRunner: OutputCommitter set in config null
19/07/05 11:33:34 INFO mapreduce.Job: Running job: job_local822758103_0001
19/07/05 11:33:34 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
19/07/05 11:33:34 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
19/07/05 11:33:34 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
19/07/05 11:33:34 INFO mapred.LocalJobRunner: Waiting for map tasks
19/07/05 11:33:34 INFO mapred.LocalJobRunner: Starting task: attempt_local822758103_0001_m_000000_0
19/07/05 11:33:34 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
19/07/05 11:33:34 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
19/07/05 11:33:34 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
19/07/05 11:33:34 INFO mapred.Task:  Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProcessTree@7bc2ab3
19/07/05 11:33:34 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/input/data.csv:0+20151734
19/07/05 11:33:34 INFO mapred.MapTask: numReduceTasks: 1
19/07/05 11:33:34 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
19/07/05 11:33:34 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
19/07/05 11:33:34 INFO mapred.MapTask: soft limit at 83886080
19/07/05 11:33:34 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
19/07/05 11:33:34 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
19/07/05 11:33:34 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
```

The map reduce operation was completed successfully

```
19/07/05 11:33:42 INFO mapred.LocalJobRunner: reduce task executor complete.
19/07/05 11:33:43 INFO mapreduce.Job:  map 100% reduce 100%
19/07/05 11:33:43 INFO mapreduce.Job: Job job_local822758103_0001 completed successfully
19/07/05 11:33:43 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=8303786
                FILE: Number of bytes written=13110521
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=40303468
                HDFS: Number of bytes written=31489
                HDFS: Number of read operations=13
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Map-Reduce Framework
                Map input records=450017
                Map output records=450017
                Map output bytes=3248855
                Map output materialized bytes=4148895
                Input split bytes=88
                Combine input records=0
                Combine output records=0
                Reduce input groups=298
                Reduce shuffle bytes=4148895
                Reduce input records=450017
                Reduce output records=900
                Spilled Records=900034
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=6
                Total committed heap usage (bytes)=662700032
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=20151734
        File Output Format Counters
                Bytes Written=31489
19/07/05 11:33:43 INFO streaming.StreamJob: Output directory: 14
```

The output is stored in output directory named “14”.
Sample output is shown

```
C:\hadoop-2.8.0>hadoop fs -cat /user/manna/14/part-00000
maximum arrival delay: ABE       786
minimum arrival delay: ABE       -30
maximum arrival delay: ABI       261
minimum arrival delay: ABI       -20
maximum arrival delay: ABQ       908
minimum arrival delay: ABQ       -47
maximum arrival delay: ABR       1245
minimum arrival delay: ABR       -29
maximum arrival delay: ABY       307
minimum arrival delay: ABY       -42
maximum arrival delay: ACT       192
minimum arrival delay: ACT       -27
maximum arrival delay: ACV       565
minimum arrival delay: ACV       -26
maximum arrival delay: ACY       649
minimum arrival delay: ACY       -43
maximum arrival delay: ADK       41
minimum arrival delay: ADK       -31
maximum arrival delay: ADQ       71
minimum arrival delay: ADQ       -31
maximum arrival delay: AEX       351
minimum arrival delay: AEX       -40
average arrival delay ABE       15.43010752688172
average arrival delay ABI       28.962962962962962
average arrival delay ABQ       5.056603773584905
average arrival delay ABR       35.03333333333333
average arrival delay ABY       3.654320987654321
average arrival delay ACT       11.142857142857142
average arrival delay ACV       10.952941176470588
average arrival delay ACY       2.139751552795031
average arrival delay ADK       2.3333333333333335
average arrival delay ADQ       -5.225
average arrival delay AEX       9.93061224489796
average arrival delay AGS       16.647058823529413
average arrival delay ALB       2.7625994694960214
average arrival delay AMA       0.8148148148148148
average arrival delay ANC       4.973176865046102
average arrival delay APN       20.904761904761905
average arrival delay ASE       32.85406301824212
average arrival delay ATL       9.904612978889757
average arrival delay ATW       25.306532663316585
average arrival delay AUS       6.0779642663779105
Top 10 airports by their average Arrival delay
LSE     -22.3
YAK     -19.3125
PPG     -12.5
CDV     -11.26923076923077
ISN     -9.619047619047619
LBE     -8.7
EAU     -8.018518518518519
ORH     -7.649122807017544
INL     -7.530612244897959
BFL     -6.788571428571428
```

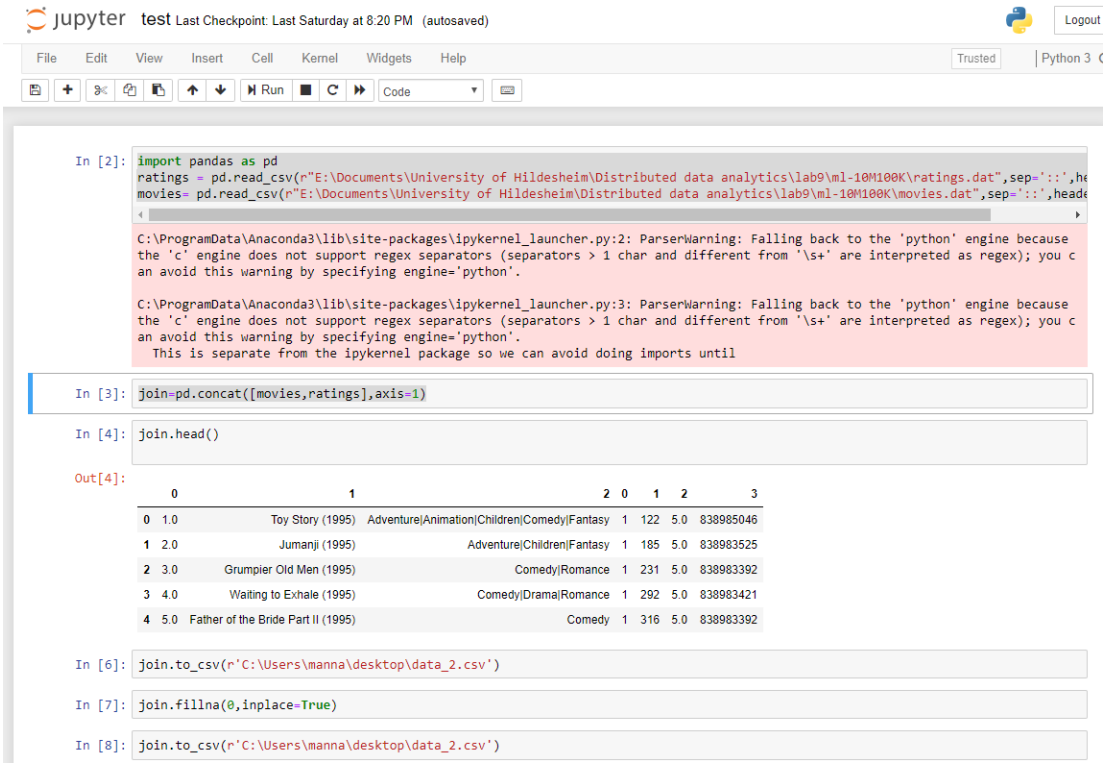
Exercise 3: Analysis of Movie dataset using Map and Reduce

The following are the steps which are being followed while doing map reduce on Airport efficiency.

1. Two directories were created in hdfs i.e input(to store input data) and output(to get output data).

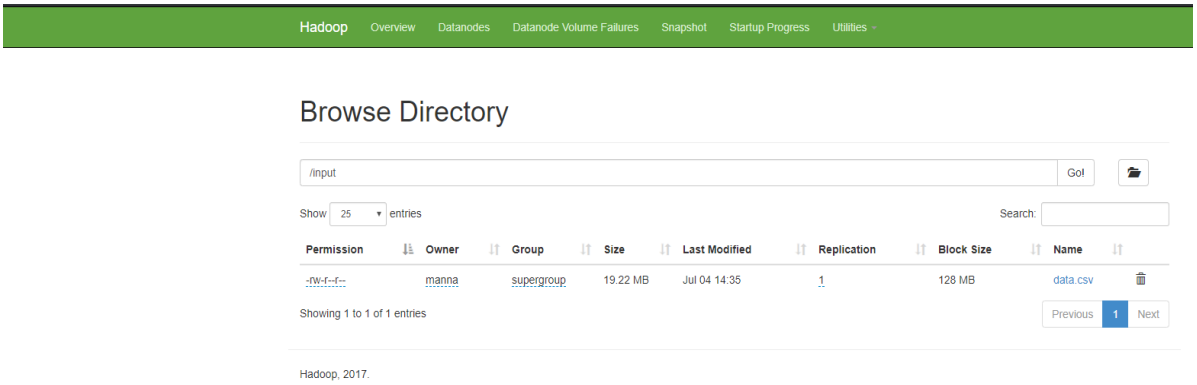
```
C:\hadoop-2.8.0\sbin>hadoop fs -mkdir /input

C:\hadoop-2.8.0\sbin>hadoop fs -mkdir /output
```

2.The two dat files(movies.dat and rating.dat) are merged to create one data.csv file



3. Uploaded the .csv file in hdfs input folder

```
C:\hadoop-2.8.0\bin>hadoop fs -put C:/Users/manna/desktop/data.csv /input
```



4. Write Mapper and Reducer in python

mapper.py code

```python
import sys

# input comes from STDIN (standard input)
movieid_name={}
for line in sys.stdin:
    line = line.strip()
    line = line.split(",")

    movie_id_map=line[0]
    movie_name=line[1]
    genre=line[2]
    user_id = line[3]
    movie_id= line[4]
    rating=line[5]
```

Reducer.py

```
#Reducer.py
import sys
import operator



movieid_name={}
movieid_rating={}
userid_rating={}
genre_rating={}
ave_movie_rating={}
ave_user_rating={}
ave_genre_rating={}

for line in sys.stdin:
    data = line.strip().split('\t')

    if len(data) != 4:
        # Something has gone wrong. Skip this line.
        continue
    movie_id_map,movie_name,genre,user_id,movie_id,rating= data

    if movie_id_map in movieid_name:
        movieid_name[movie_id_map].append(int(movie_name))
    else:
        movieid_name[movie_id_map] = []
        movieid_name[movie_id_map].append(int(movie_name))

    if movie_id in movieid_rating:
        movieid_rating[movie_id].append(int(rating))
    else:
        movieid_rating[movie_id] = []
        movieid_rating[movie_id].append(int(rating))

    if user_id in userid_rating:
        userid_rating[user_id].append(int(rating))
    else:
        userid_rating[user_id] = []
        userid_rating[user_id].append(int(rating))

    if genre in genre_rating:
        genre_rating[genre].append(int(rating))
    else:
        genre_rating[genre] = []
        genre_rating[genre].append(int(rating))


        
for movie_id in movieid_rating.keys():
    ave_movie_rating[movieid_name[movie_id]].append(sum(movieid_rating[movie_id])*1.0 / len(movieid_rating[movie_id]))


for user_id in userid_rating.keys():
    ave_user_rating[user_id].append(sum(userid_rating[user_id])*1.0 / len(userid_rating[user_id]))
    if userid_rating[user_id]>40:
        target_user=user_id

for genre in genre_rating.keys():
    ave_genre_rating[genre_rating[genre]].append(sum(genre_rating[genre])*1.0 / len(genre_rating[genre]))


print(max(ave_movie_rating.iteritems(), key=operator.itemgetter(1))[0])
print(min(ave_user_rating[user_id].iteritems(), key=operator.itemgetter(1))[0])
print(max(ave_genre_rating[genre].iteritems(), key=operator.itemgetter(1))[0])
```

4.1 Explaining the reducer.py code

Collecting data line by line by strip and split function as well as performed a quick check if I have data of length is not equals to 6.If the data is correct then the 'data' is unpacked and saved in movie_id_map,movie_name,genre,user_id,movie_id,rating respectively

```
for line in sys.stdin:
    data = line.strip().split('\t')

    if len(data) != 6:
        # Something has gone wrong. Skip this line.
        continue
    movie_id_map,movie_name,genre,user_id,movie_id,rating= data
```

This if else is used to store the movie name according to the movie_id in a dictionary 'movieid_name'

```
    if movie_id_map in movieid_name:
        movieid_name[movie_id_map].append(int(movie_name))
    else:
        movieid_name[movie_id_map] = []
        movieid_name[movie_id_map].append(int(movie_name))
```

This if else is used to store rating of movie according to their respective movie_id in a dictionary 'movieid_rating'

```
    if movie_id in movieid_rating:
        movieid_rating[movie_id].append(int(rating))
    else:
        movieid_rating[movie_id] = []
        movieid_rating[movie_id].append(int(rating))
```

This if else is used to store rating of movie according to the user_id who rated in a dictionary 'userid_rating'

```
if user_id in userid_rating:
    userid_rating[user_id].append(int(rating))
else:
    userid_rating[user_id] = []
    userid_rating[user_id].append(int(rating))
```

This if else is used to store rating according to genre in a dictionary ‘genre_rating’

```
if genre in genre_rating:
    genre_rating[genre].append(int(rating))
else:
    genre_rating[genre] = []
    genre_rating[genre].append(int(rating))
```

This for loop find the average rating for each movie id and in the print function the maximum value is collected then the key(movie_name) is printed which has the maximum average rating.

```
for movie_id in movieid_rating.keys():
    ave_movie_rating[movieid_name[movie_id]].append(sum(movieid_rating[movie_id])*1.0 / len(movieid_rating[movie_id]))
print(max(ave_movie_rating.iteritems(), key=operator.itemgetter(1))[0])
```

This for loop finds the average rating according to each user but the user has to rate at least 40 times hence a condition check is done whether the length is greater than 40 or not.If it is greater the key and value is stored in ave_user_rating dictionary.At the end the user_id is printed who has given the minimum rating for at least 40 times.

```
for user_id in userid_rating.keys():
    if(len(userid_rating[user_id]>40)):
        ave_user_rating[user_id].append(sum(userid_rating[user_id])*1.0 / len(userid_rating[user_id]))

print(min(ave_user_rating.iteritems(), key=operator.itemgetter(1))[0])
```

This loop is used to find the average rating for each genre and later the genre is printed which has the maximum rating.

```
for genre in genre_rating.keys():
    ave_genre_rating[genre_rating[genre]].append(sum(genre_rating[genre])*1.0 / len(genre_rating[genre]))

print(max(ave_genre_rating[genre].iteritems(), key=operator.itemgetter(1))[0])
```

6. Testing mapper and reducer script in Hadoop platform

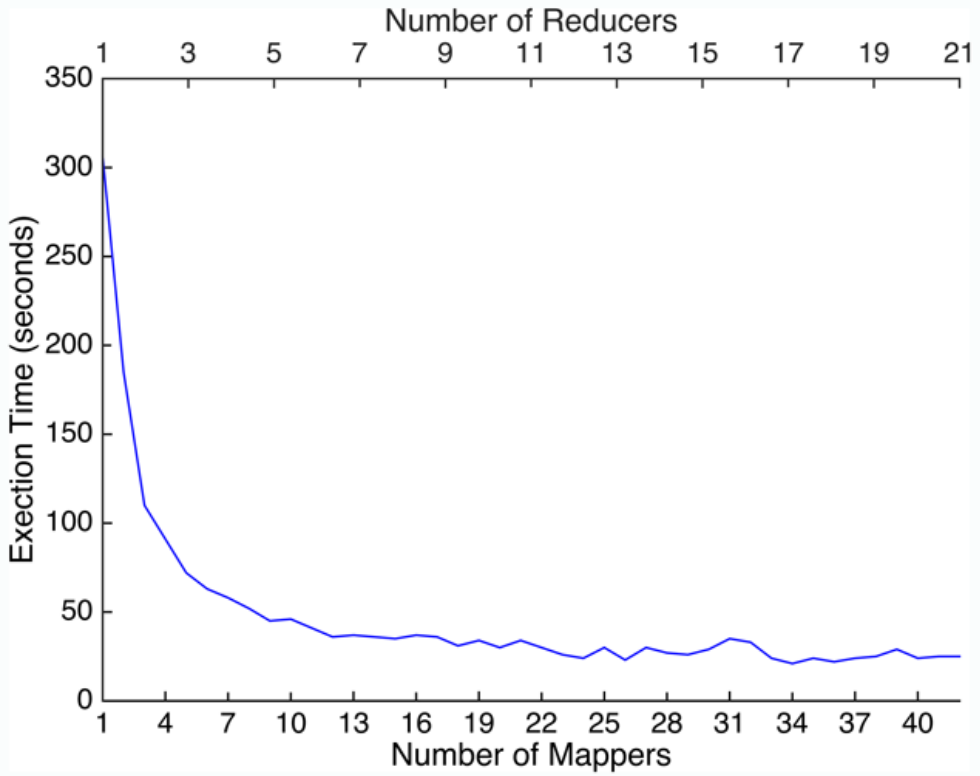
Using Hadoop streaming-2.8.0 jar to execute the mapper and reducer script
Command "**hadoop jar C:\hadoop-2.8.0\share\hadoop\tools\lib\hadoop-streaming-2.8.0.jar -file mapper.py -mapper "python mapper.py" -file reducer.py -reducer "python reducer.py" -input /input/data_2.csv -output 14**"

```
C:\hadoop-2.8.0>hadoop jar C:\hadoop-2.8.0\share\hadoop\tools\lib\hadoop-streaming-2.8.0.jar -file mapper.py -mapper "python mapper.py" -file reducer.py -reducer "python reducer.py" -input /input/data.csv -output 14
19/07/05 11:33:31 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper.py, reducer.py] [] C:\Users\manna\AppData\Local\Temp\streamjob2840630007336944374.jar tmpDir=null
19/07/05 11:33:32 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
19/07/05 11:33:32 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
19/07/05 11:33:32 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
19/07/05 11:33:33 INFO mapred.FileInputFormat: Total input files to process : 1
19/07/05 11:33:33 INFO mapreduce.JobSubmitter: number of splits:1
19/07/05 11:33:33 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local822758103_0001
19/07/05 11:33:34 INFO mapred.LocalDistributedCacheManager: Localized file:/C:/hadoop-2.8.0/mapper.py as file:/tmp/hadoop-manna/mapred/local/1562319214005/mapper.py
19/07/05 11:33:34 INFO mapred.LocalDistributedCacheManager: Localized file:/C:/hadoop-2.8.0/reducer.py as file:/tmp/hadoop-manna/mapred/local/1562319214006/reducer.py
19/07/05 11:33:34 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
19/07/05 11:33:34 INFO mapred.LocalJobRunner: OutputCommitter set in config null
19/07/05 11:33:34 INFO mapreduce.Job: Running job: job_local822758103_0001
19/07/05 11:33:34 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
19/07/05 11:33:34 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
19/07/05 11:33:34 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
19/07/05 11:33:34 INFO mapred.LocalJobRunner: Waiting for map tasks
19/07/05 11:33:34 INFO mapred.LocalJobRunner: Starting task: attempt_local822758103_0001_m_000000_0
19/07/05 11:33:34 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
19/07/05 11:33:34 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
19/07/05 11:33:34 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
19/07/05 11:33:34 INFO mapred.Task:  Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProcessTree@7bc2ab3
19/07/05 11:33:34 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/input/data.csv:0+20151734
19/07/05 11:33:34 INFO mapred.MapTask: numReduceTasks: 1
19/07/05 11:33:34 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
19/07/05 11:33:34 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
19/07/05 11:33:34 INFO mapred.MapTask: soft limit at 83886080
19/07/05 11:33:34 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
19/07/05 11:33:34 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
19/07/05 11:33:34 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
```

The map reduce operation was completed successfully

```
19/07/05 11:33:42 INFO mapred.LocalJobRunner: reduce task executor complete.
19/07/05 11:33:43 INFO mapreduce.Job:  map 100% reduce 100%
19/07/05 11:33:43 INFO mapreduce.Job: Job job_local822758103_0001 completed successfully
19/07/05 11:33:43 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=8303786
                FILE: Number of bytes written=13110521
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=40303468
                HDFS: Number of bytes written=31489
                HDFS: Number of read operations=13
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Map-Reduce Framework
                Map input records=450017
                Map output records=450017
                Map output bytes=3248855
                Map output materialized bytes=4148895
                Input split bytes=88
                Combine input records=0
                Combine output records=0
                Reduce input groups=298
                Reduce shuffle bytes=4148895
                Reduce input records=450017
                Reduce output records=900
                Spilled Records=900034
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=6
                Total committed heap usage (bytes)=662700032
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=20151734
        File Output Format Counters
                Bytes Written=31489
19/07/05 11:33:43 INFO streaming.StreamJob: Output directory: 14
```

The graph of execution time vs the number of mappers × reducers for the overall task



The output is stored in output directory named “14”.
Sample output is shown.

```
Follow the Bitch(1998)
 ('3598',1.0153846153846153)
 ('Animation|Comedy|Thriller',4.473837209302325)
```