



PROJECT 1: Who is Most Popular?

DS503
Big Data Management
Team 1

Author:
Han Jiang
Xiaoting Cui
Dennis Silva

Supervisor:
Dr.Rundensteiner

September 22, 2017

Preface:

The following code was written and ran within the already-built vm provided by matabakh.

Note: Make sure you start hadoop first!

CREATE A HDFS DIRECTORY

if you want to make a new directory to put code in use:

```
>> hadoop dfs -mkdir <new_path>
```

for example:

```
>> hadoop dfs -mkdir /user/hadoop/pigdata
```

DOWNLOAD OUTPUT

once you complete a job and store your output, to download output file stored in hdfs to local system use:

```
>> hadoop dfs -get /user/hadoop/<output_file_name> <local_path>
```

for example:

```
>> hadoop dfs -get /user/hadoop/sampleOutput /home/hadoop/Documents/Project1-Pig
```

ADD A FILE/FOLDER

to add a file to the hdfs (file to be read later)

```
>> sudo hadoop dfs -put <input_file_path> <hdfs_output_file_path>
```

for example:

```
>> sudo hadoop dfs -put /home/hadoop/Downloads/sampleFriends.csv /pigdata
```

DELETE A FILE/FOLDER

to delete/remove a file/folder from hdfs:

```
>> hadoop fs -rmr <remove_directory_folder>
```

for example:

```
>> hadoop fs -rmr /user/hadoop/sampleOutput
```

Hadoop variable:

When we want to write Int (float, text) into hadoop, we use IntWritable (FloatWritable, TextWritable).

Combiner:

The general idea of combiner is to have local reducer which can reduce the shuffle task for hadoop, resulting high efficiency. However, when some node is busy, some mapper will skip the combiner process, directly go to reducer. In this way, combiner should have the same input format and output format, both same as mapper output. When just summing up, it is good to have combiner, but in other cases, combiner may not be suitable.

1. Creating Datasets

Java Program: AccessLog.java, AccessLogWriter.java, Friends.java, FriendsWriter.java, MyPage.java, MyPageWriter.java, CreateData.java

AccessLog.java, Friends.java, MyPage.java are used to generate 3 different classes required in this task.; AccessLogWriter.java, FriendsWriter.java, and MyPageWriter.java are used to define 3 tables, and CreateData.java is to create 3 datasets.

Go to the directory containing the code:

```
>mkdir task1  
>javac -classpath /usr/share/hadoop/hadoop-core-1.2.1.jar -d  
task1 ./AccessLog.java ./AccessLogWriter.java ./Friends.java ./FriendsWriter.java ./MyPage.jav  
a ./MyPageWriter.java ./createData.java
```

Go to task1 which containing class

```
>cd task1
```

Execute

```
>java createData
```

Then, we can get our three datasets: MyPage, Friends, and AccessLog

2. Loading Datasets to Hadoop

Create new directory:

```
> hadoop fs -mkdir /user/hadoop/Proj1/data
```

Upload data:

```
> cd  
> hadoop fs -put proj1/javacode/task1/Friends.csv /user/hadoop/Proj1/data  
> hadoop fs -put proj1/javacode/task1/MyPage.csv /user/hadoop/Proj1/data  
> hadoop fs -put proj1/javacode/task1/AccessLog.csv /user/hadoop/Proj1/data
```

Contents of directory [/user/hadoop/Proj1/data](#)

Goto :

[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
AccessLog.csv	file	576 MB	3	64 MB	2017-09-21 18:26	rw-r--r--	hadoop	hadoop
Friends.csv	file	1.22 GB	3	64 MB	2017-09-21 18:25	rw-r--r--	hadoop	hadoop
MyPage.csv	file	5.34 MB	3	64 MB	2017-09-21 18:26	rw-r--r--	hadoop	hadoop

[Go back to DFS home](#)

3. Accomplishing Analytics Tasks using MapReduce Jobs

task3a.)

We only use one mapper to read MyPage dataset line by line, generating key(ID)-value(Nationality). And see if Nationality is equal to what we have chosen.

We did not use combiner because we do no need to put things together.

-- sample execution:

```
> hadoop jar ./project1.jar my.twoa /user/hadoop/Proj1/data/MyPage.csv  
/user/hadoop/Proj1/javacode_a
```

Result:

Goto : /user/hadoop/Proj1/javacode_a

[Go back to dir listing](#)

[Advanced view/download options](#)

```
FdgRRD1Mx7zon, GWgJOME0v7DjtK1  
|
```

For this task, it used 210 ms.

```
17/09/22 11:31:36 INFO mapred.JobClient: FileSystemCounters  
17/09/22 11:31:36 INFO mapred.JobClient: HDFS_BYTES_READ=5599726  
17/09/22 11:31:36 INFO mapred.JobClient: FILE_BYTES_WRITTEN=55295  
17/09/22 11:31:36 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=30  
17/09/22 11:31:36 INFO mapred.JobClient: File Input Format Counters  
17/09/22 11:31:36 INFO mapred.JobClient: Bytes Read=5599606  
17/09/22 11:31:36 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 11:31:36 INFO mapred.JobClient: Map input records=100000  
17/09/22 11:31:36 INFO mapred.JobClient: Physical memory (bytes) snapshot=43  
245568  
17/09/22 11:31:36 INFO mapred.JobClient: Spilled Records=0  
17/09/22 11:31:36 INFO mapred.JobClient: CPU time spent (ms)=210  
17/09/22 11:31:36 INFO mapred.JobClient: Total committed heap usage (bytes)=  
16252928  
17/09/22 11:31:36 INFO mapred.JobClient: Virtual memory (bytes) snapshot=347  
430912  
17/09/22 11:31:36 INFO mapred.JobClient: Map output records=1
```

Task3b.)

We use one mapper to read MyPage dataset line by line, generating key(CountryCode)-value(1) pairs. And one reducer takes the output key-value pairs from mapper to group by the CountryCode, counting how many persons are there.

We can use combiner, because we only need reducer to sum up, and when using combiner, it is faster.

With combiner:

-- sample execution:

```
> hadoop jar ./project1.jar my.twob /user/hadoop/Proj1/data/MyPage.csv  
/user/hadoop/Proj1/javacode_b
```

Result:

Goto : [/user/hadoop/Proj1/javacode_b](#)

[Go back to dir listing](#)

[Advanced view/download options](#)

```
1      9860  
10     9970  
2      10026  
3      10058  
4      10117  
5      10110  
6      10083  
7      9919  
8      9815  
9      10042
```

In this task, it used 610ms.

```
17/09/22 11:34:14 INFO mapred.JobClient: File Input Format Counters  
17/09/22 11:34:14 INFO mapred.JobClient: Bytes Read=5599606  
17/09/22 11:34:14 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 11:34:14 INFO mapred.JobClient: Map output materialized bytes=87  
17/09/22 11:34:14 INFO mapred.JobClient: Map input records=100000  
17/09/22 11:34:14 INFO mapred.JobClient: Reduce shuffle bytes=87  
17/09/22 11:34:14 INFO mapred.JobClient: Spilled Records=20  
17/09/22 11:34:14 INFO mapred.JobClient: Map output bytes=609970  
17/09/22 11:34:14 INFO mapred.JobClient: Total committed heap usage (bytes)=  
132190208  
17/09/22 11:34:14 INFO mapred.JobClient: CPU time spent (ms)=610  
17/09/22 11:34:14 INFO mapred.JobClient: Combine input records=100000  
17/09/22 11:34:14 INFO mapred.JobClient: SPLIT_RAW_BYTES=120  
17/09/22 11:34:14 INFO mapred.JobClient: Reduce input records=10  
17/09/22 11:34:14 INFO mapred.JobClient: Reduce input groups=10  
17/09/22 11:34:14 INFO mapred.JobClient: Combine output records=10  
17/09/22 11:34:14 INFO mapred.JobClient: Physical memory (bytes) snapshot=18  
7875328  
17/09/22 11:34:14 INFO mapred.JobClient: Reduce output records=10
```

Without combiner:

-- sample execution:

```
> hadoop jar ./project1.jar my.twob_nocombiner /user/hadoop/Proj1/data/MyPage.csv  
/user/hadoop/Proj1/javacode_b_nc
```

Result:

Goto : [/user/hadoop/Proj1/javacode_b_nc](#)

[Go back to dir listing](#)

[Advanced view/download options](#)

```
1      9860  
10     9970  
2      10026  
3      10058  
4      10117  
5      10110  
6      10083  
7      9919  
8      9815  
9      10042
```

For this task, it used 690ms.

```
17/09/22 11:38:41 INFO mapred.JobClient:      Map input records=100000  
17/09/22 11:38:41 INFO mapred.JobClient:      Reduce shuffle bytes=809976  
17/09/22 11:38:41 INFO mapred.JobClient:      Spilled Records=200000  
17/09/22 11:38:41 INFO mapred.JobClient:      Map output bytes=609970  
17/09/22 11:38:41 INFO mapred.JobClient:      Total committed heap usage (bytes)=  
132190208  
17/09/22 11:38:41 INFO mapred.JobClient:      CPU time spent (ms)=690  
17/09/22 11:38:41 INFO mapred.JobClient:      Combine input records=0  
17/09/22 11:38:41 INFO mapred.JobClient:      SPLIT_RAW_BYTES=120  
17/09/22 11:38:41 INFO mapred.JobClient:      Reduce input records=100000  
17/09/22 11:38:41 INFO mapred.JobClient:      Reduce input groups=10  
17/09/22 11:38:41 INFO mapred.JobClient:      Combine output records=0  
17/09/22 11:38:41 INFO mapred.JobClient:      Physical memory (bytes) snapshot=19  
2991232  
17/09/22 11:38:41 INFO mapred.JobClient:      Reduce output records=10
```

Task3c.)

We use one mapper to read AccessLog dataset line by line, generation key(WhatPage)-value(1) pairs. And one reducer takes the output key-value pairs from mapper to group by WhatPage, using PriorityQuene and Comparator to sort and return top 10 values.

We cannot use combiner, because some of the mapper may not use combiner before reducer, resulting the hashtable being strange.

-- sample execution:

```
> hadoop jar ./task3c.jar task2c /user/hadoop/Proj1/data/AccessLog.csv  
/user/hadoop/Proj1/javacode_c1
```

Without combiner:

Result:

Goto : [/user/hadoop/Proj1/javacode_c1](#)

[Go back to dir listing](#)

[Advanced view/download options](#)

```
59666 137  
24271 137  
37883 137  
63713 136  
33985 136  
62485 136  
38339 136  
38048 136  
79952 135  
54706 135
```

For this task, it used 38760ms.

```
17/09/22 14:16:42 INFO mapred.JobClient: File Input Format Counters  
17/09/22 14:16:42 INFO mapred.JobClient: Bytes Read=604012544  
17/09/22 14:16:42 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 14:16:42 INFO mapred.JobClient: Map output materialized bytes=96554  
364  
17/09/22 14:16:42 INFO mapred.JobClient: Map input records=9655431  
17/09/22 14:16:42 INFO mapred.JobClient: Reduce shuffle bytes=96554364  
17/09/22 14:16:42 INFO mapred.JobClient: Spilled Records=28966293  
17/09/22 14:16:42 INFO mapred.JobClient: Map output bytes=77243448  
17/09/22 14:16:42 INFO mapred.JobClient: Total committed heap usage (bytes)=  
1590714368  
17/09/22 14:16:42 INFO mapred.JobClient: CPU time spent (ms)=38760  
17/09/22 14:16:42 INFO mapred.JobClient: Combine input records=0  
17/09/22 14:16:42 INFO mapred.JobClient: SPLIT_RAW_BYTES=1107  
17/09/22 14:16:42 INFO mapred.JobClient: Reduce input records=9655431  
17/09/22 14:16:42 INFO mapred.JobClient: Reduce input groups=100000  
17/09/22 14:16:42 INFO mapred.JobClient: Combine output records=0  
17/09/22 14:16:42 INFO mapred.JobClient: Physical memory (bytes) snapshot=18  
08633856
```

Task3d.)

We use two mappers. One is to read MyPage line by line, generating key(ID)-value(Name) pairs with a file flag "M". The other one is to read Friends dataset line by line, generating key(Friend)-value(1) pairs with a file flag "F". And one reducer takes the output from these mappers(join these two datasets by person id) to count the number of people listing this user as a friend.

We cannot use combiner, because it is not a sum up job. For MyPage dataset, we need to get Name, while for Friends we count the total number. Because when the node is busy, they will skip combiner.

-- sample execution:

```
> hadoop jar ./project1.jar my.twod_nocombiner /user/hadoop/Proj1/data/MyPage.csv  
/user/hadoop/Proj1/data/Friends.csv /user/hadoop/Proj1/javacode_d
```

Without combiner:

Partial result:

Goto : [/user/hadoop/Proj1/javacode_d](#) go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

```
1 BgqDsJuHHTqkjw7,198  
2 3rLrDcVHl1xVtCjy,196  
3 2Tks3zDt0kB5GI,201  
4 UXWIBVJ1xaBRbFrHHY1,186  
5 QlLHvey2lKEQsJ,194  
6 FdgRD1Mx7zon,193  
7 dl90No95NMRo,183  
8 7byM05A0kRvyp4r4MLYD,206  
9 6hv1jvU30NK3iii,213  
10 DjA82pSuFFc0I,219  
11 SgXYPXRp4iBm,203  
12 encvGrQ3hI,213  
13 PlOKWm0gcq,216  
14 Of7SGFjY4zgfGYIO,175  
15 MynH22rEgwoS3Rqghh0C,195  
16 BgSjuWeVTY6EGa,193  
17 xYR7vPhfN0enC,208  
18 VvyihVgWGkNVUP,208  
19 Zpm0Dpdmhle8Qabtx2q,199  
20 q71fReLb83hZS7GVng,211  
21 37UWMSaC1GLSEAWx,189  
22 UfQSd4ySagMY8Fvdl,212  
23 AzyPVPiCiwuVX7XxN17g,193
```

For this task, used 251710ms.

```
17/09/22 12:00:05 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 12:00:05 INFO mapred.JobClient: Map output materialized bytes=28280  
0325  
17/09/22 12:00:05 INFO mapred.JobClient: Map input records=20100000  
17/09/22 12:00:05 INFO mapred.JobClient: Reduce shuffle bytes=282800325  
17/09/22 12:00:05 INFO mapred.JobClient: Spilled Records=60200000  
17/09/22 12:00:05 INFO mapred.JobClient: Map output bytes=242600199  
17/09/22 12:00:05 INFO mapred.JobClient: Total committed heap usage (bytes)=  
3533983744  
17/09/22 12:00:05 INFO mapred.JobClient: CPU time spent (ms)=251710  
17/09/22 12:00:05 INFO mapred.JobClient: Combine input records=0  
17/09/22 12:00:05 INFO mapred.JobClient: SPLIT_RAW_BYTES=5438  
17/09/22 12:00:05 INFO mapred.JobClient: Reduce input records=20100000  
17/09/22 12:00:05 INFO mapred.JobClient: Reduce input groups=100000  
17/09/22 12:00:05 INFO mapred.JobClient: Combine output records=0  
17/09/22 12:00:05 INFO mapred.JobClient: Physical memory (bytes) snapshot=41  
93480704  
17/09/22 12:00:05 INFO mapred.JobClient: Reduce output records=100000  
17/09/22 12:00:05 INFO mapred.JobClient: Virtual memory (bytes) snapshot=765  
2909056  
17/09/22 12:00:05 INFO mapred.JobClient: Map output records=20100000
```

Task3e.)

We use one mapper to read AccessLog dataset line by line, generating key(ByWho)-value(WhatPage) pairs. And one reducer takes these output, creating a hashset to store unique WhatPage and a int value to store total number of WhatPage for each key.

We cannot use combiner, because we have to find the unique value.

-- sample execution:

```
> hadoop jar ./project1.jar my.twoe_nocombiner /user/hadoop/Proj1/data/AccessLog.csv  
/user/hadoop/Proj1/javacode_e
```

Without combiner:

Partial result:

Goto : [/user/hadoop/Proj1/javacode_e](#) go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

```
1    88,88  
2    78,78  
3    99,99  
4    98,98  
5    101,101  
6    107,107  
7    103,103  
8    100,100  
9    106,106  
10   93,93  
11   94,94  
12   96,96  
13   94,94  
14   91,91  
15   105,105  
16   87,87  
17   88,88  
18   80,80  
19   86,86  
20   86,86  
21   105,105  
22   99,100  
23   97,97  
24   108,108  
25   102,102  
26   97,97
```

For this task, it used 51460ms.

```
17/09/22 12:30:31 INFO mapred.JobClient: File Input Format Counters  
17/09/22 12:30:31 INFO mapred.JobClient: Bytes Read=604012544  
17/09/22 12:30:31 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 12:30:31 INFO mapred.JobClient: Map output materialized bytes=15341  
4904  
17/09/22 12:30:31 INFO mapred.JobClient: Map input records=9655431  
17/09/22 12:30:31 INFO mapred.JobClient: Reduce shuffle bytes=153414904  
17/09/22 12:30:31 INFO mapred.JobClient: Spilled Records=28966293  
17/09/22 12:30:31 INFO mapred.JobClient: Map output bytes=134103988  
17/09/22 12:30:31 INFO mapred.JobClient: Total committed heap usage (bytes)=  
1215594496  
17/09/22 12:30:31 INFO mapred.JobClient: CPU time spent (ms)=51460  
17/09/22 12:30:31 INFO mapred.JobClient: Combine input records=0  
17/09/22 12:30:31 INFO mapred.JobClient: SPLIT_RAW_BYTES=1107  
17/09/22 12:30:31 INFO mapred.JobClient: Reduce input records=9655431  
17/09/22 12:30:31 INFO mapred.JobClient: Reduce input groups=100000  
17/09/22 12:30:31 INFO mapred.JobClient: Combine output records=0  
17/09/22 12:30:31 INFO mapred.JobClient: Physical memory (bytes) snapshot=15  
22278400  
17/09/22 12:30:31 INFO mapred.JobClient: Reduce output records=100000
```

Task3f.)

We use two mappers. One is to read Friends dataset line by line, generating key(PersonID)-value(MyFriend) pairs with a file flag “F”. The other one is to read AccessLog dataset line by line, generating key(ByWho)-value(WhatPage) pairs with a file flag “A”. And one reducer takes all the output, grouping by PersonID, creating two hashsets to store unique friend id(MyFriend) and page id(WhatPage). If there is a friend of a certain user never appears in the WhatPage set, then we can specify this user declared someone as his friend yet never accessed his friend’s page.

We cannot use combiner, because this is not a sum up job. We have to look at all the mappers not only the local one.

-- sample execution:

```
> hadoop jar ./project1.jar my.twof_nocombiner /user/hadoop/Proj1/data/Friends.csv  
/user/hadoop/Proj1/data/AccessLog.csv /user/hadoop/Proj1/javacode_f
```

Without combiner:

Partial result:

Goto : [/user/hadoop/Proj1/javacode_f](#)

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

```
1 this person has a friends never visited  
2 this person has a friends never visited  
3 this person has a friends never visited  
4 this person has a friends never visited  
5 this person has a friends never visited  
6 this person has a friends never visited  
7 this person has a friends never visited  
8 this person has a friends never visited  
9 this person has a friends never visited  
10 this person has a friends never visited  
11 this person has a friends never visited  
12 this person has a friends never visited  
13 this person has a friends never visited  
14 this person has a friends never visited  
15 this person has a friends never visited  
16 this person has a friends never visited  
17 this person has a friends never visited  
18 this person has a friends never visited  
19 this person has a friends never visited  
20 this person has a friends never visited  
21 this person has a friends never visited
```

For this task, it used 178080ms.

```
17/09/22 12:39:14 INFO mapred.JobClient: File Input Format Counters  
17/09/22 12:39:14 INFO mapred.JobClient: Bytes Read=0  
17/09/22 12:39:14 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 12:39:14 INFO mapred.JobClient: Map output materialized bytes=53050  
6902  
17/09/22 12:39:14 INFO mapred.JobClient: Map input records=29655431  
17/09/22 12:39:14 INFO mapred.JobClient: Reduce shuffle bytes=530506902  
17/09/22 12:39:14 INFO mapred.JobClient: Spilled Records=88966293  
17/09/22 12:39:14 INFO mapred.JobClient: Map output bytes=471195866  
17/09/22 12:39:14 INFO mapred.JobClient: Total committed heap usage (bytes)=  
4775268352  
17/09/22 12:39:14 INFO mapred.JobClient: CPU time spent (ms)=178080  
17/09/22 12:39:14 INFO mapred.JobClient: Combine input records=0  
17/09/22 12:39:14 INFO mapred.JobClient: SPLIT_RAW_BYTES=7529  
17/09/22 12:39:14 INFO mapred.JobClient: Reduce input records=29655431  
17/09/22 12:39:14 INFO mapred.JobClient: Reduce input groups=100000  
17/09/22 12:39:14 INFO mapred.JobClient: Combine output records=0  
17/09/22 12:39:14 INFO mapred.JobClient: Physical memory (bytes) snapshot=56  
73529344  
17/09/22 12:39:14 INFO mapred.JobClient: Reduce output records=100000
```

Task3g.)

We use one mapper to read AccessLog dataset line by line, generating key(ByWho)-value(AccessTime) pairs. And one reducer takes the output, grouping by ByWho. In the reducer, we find the last visit time(maximum number in AccessTime) and initialize a time unit, finding out whoes last visit is before that day.

We cannot use combiner, we have to look at all the mappers to find the global maximum.

-- sample execution:

```
> hadoop jar ./project1.jar my.twog_nocombiner /user/hadoop/Proj1/data/AccessLog.csv  
/user/hadoop/Proj1/javacode_g
```

Without combiner:

Result:

Goto : [/user/hadoop/Proj1/javacode_g](#)

[Go back to dir listing](#)

[Advanced view/download options](#)

```
7078 This person lost interest  
17132 This person lost interest  
28202 This person lost interest  
30016 This person lost interest  
40552 This person lost interest  
41101 This person lost interest  
42486 This person lost interest  
48304 This person lost interest  
69346 This person lost interest  
73861 This person lost interest  
89929 This person lost interest  
94441 This person lost interest  
95032 This person lost interest  
96832 This person lost interest
```

For this task, it used 49470ms.

```
17/09/22 12:51:20 INFO mapred.JobClient: File Input Format Counters  
17/09/22 12:51:20 INFO mapred.JobClient: Bytes Read=604012544  
17/09/22 12:51:20 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 12:51:20 INFO mapred.JobClient: Map output materialized bytes=163069936  
17/09/22 12:51:20 INFO mapred.JobClient: Map input records=9655431  
17/09/22 12:51:20 INFO mapred.JobClient: Reduce shuffle bytes=163069936  
17/09/22 12:51:20 INFO mapred.JobClient: Spilled Records=28966293  
17/09/22 12:51:20 INFO mapred.JobClient: Map output bytes=143759020  
17/09/22 12:51:20 INFO mapred.JobClient: Total committed heap usage (bytes)=1225908224  
17/09/22 12:51:20 INFO mapred.JobClient: CPU time spent (ms)=49470  
17/09/22 12:51:20 INFO mapred.JobClient: Combine input records=0  
17/09/22 12:51:20 INFO mapred.JobClient: SPLIT_RAW_BYTES=1107  
17/09/22 12:51:20 INFO mapred.JobClient: Reduce input records=9655431  
17/09/22 12:51:20 INFO mapred.JobClient: Reduce input groups=100000  
17/09/22 12:51:20 INFO mapred.JobClient: Combine output records=0  
17/09/22 12:51:20 INFO mapred.JobClient: Physical memory (bytes) snapshot=1521668096  
17/09/22 12:51:20 INFO mapred.JobClient: Reduce output records=14
```

Task3h.)

We use one mapper to read Friends dataset line by line, generating key(PersonID)-value(1) pairs. And one reducer gets average number of friends for all the users and that user's actual number of friends by reading in mapper output. We create a hashtable to store key(personID)-value(number of friends).

We cannot use combiner, because some mapper will skip combiner. In this way, the total number of friends for all user and the number of user will be wrong.

-- sample execution:

```
> hadoop jar ./project1.jar my.twoh /user/hadoop/Proj1/data/Friends.csv  
/user/hadoop/Proj1/javacode_h1
```

Partial Report:

Goto : [/user/hadoop/Proj1/javacode_h1](#) go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

```
99999 This person is popular  
99998 This person is popular  
99997 This person is popular  
99996 This person is popular  
99994 This person is popular  
99993 This person is popular  
99992 This person is popular  
99991 This person is popular  
99990 This person is popular  
99989 This person is popular  
99988 This person is popular  
99986 This person is popular  
99982 This person is popular  
99976 This person is popular  
99974 This person is popular  
99973 This person is popular  
99970 This person is popular  
99969 This person is popular
```

For this task, it used 103120ms.

```
17/09/22 15:39:37 INFO mapred.JobClient: File Input Format Counters  
17/09/22 15:39:37 INFO mapred.JobClient: Bytes Read=1312307332  
17/09/22 15:39:37 INFO mapred.JobClient: Map-Reduce Framework  
17/09/22 15:39:37 INFO mapred.JobClient: Map output materialized bytes=26000  
0120  
17/09/22 15:39:37 INFO mapred.JobClient: Map input records=20000000  
17/09/22 15:39:37 INFO mapred.JobClient: Reduce shuffle bytes=260000120  
17/09/22 15:39:37 INFO mapred.JobClient: Spilled Records=60000000  
17/09/22 15:39:37 INFO mapred.JobClient: Map output bytes=220000000  
17/09/22 15:39:37 INFO mapred.JobClient: Total committed heap usage (bytes)=  
2521513984  
17/09/22 15:39:37 INFO mapred.JobClient: CPU time spent (ms)=103120  
17/09/22 15:39:37 INFO mapred.JobClient: Combine input records=0  
17/09/22 15:39:37 INFO mapred.JobClient: SPLIT_RAW_BYTES=2420  
17/09/22 15:39:37 INFO mapred.JobClient: Reduce input records=20000000  
17/09/22 15:39:37 INFO mapred.JobClient: Reduce input groups=100000  
17/09/22 15:39:37 INFO mapred.JobClient: Combine output records=0  
17/09/22 15:39:37 INFO mapred.JobClient: Physical memory (bytes) snapshot=31  
50733312
```

4. Accomplishing Analytics Tasks Apache Pig

task4a.)

Loads in your generated MyPage data and first filters only the nationality specified. While filtering, it only stores name and hobby to then write back to an output file.

-- sample execution:

```
>pig -f task4a.pig -param MyPage=/user/hadoop/Proj1/data/MyPage.csv -param nationality=LDPzl2fYhucyXJ -param output=sampleOutput
```

Result:

File: [/user/hadoop/sampleOutput/part-m-00000](#)

Goto : [/user/hadoop/sampleOutput](#)

[Go back to dir listing](#)

[Advanced view/download options](#)

```
FdgRRD1Mx7zon  GWgJOME0v7DjtK1
```

For task a, used 14 second.

```
HadoopVersion  PigVersion      UserId  StartedAt      FinishedAt      Features
1.2.1  0.13.0  hadoop  2017-09-21 23:54:55  2017-09-21 23:55:09  FILTER

Success!

Job Stats (time in seconds):
JobId  Maps    Reduces   MaxMapTime      MinMapTime      AvgMapTime      MedianMa
pTime  MaxReduceTime  MinReduceTime  AvgReduceTime  MedianReducetime  A
lias  Feature Outputs
job_201709212003_0030  1        0          2            2            2            2            0            0
0        0        data,name_hobby,nat_filt      MAP_ONLY      hdfs://localhost
:8020/user/hadoop/sampleOutput,

Input(s):
Successfully read 100000 records (5599980 bytes) from: "/user/hadoop/Proj1/data/
MyPage.csv"

Output(s):
Successfully stored 1 records (30 bytes) in: "hdfs://localhost:8020/user/hadoop/
sampleOutput"
```

task4b.)

Loads in your generated MyPage data and groups all data by country code while counting the number of pages (rows) that have that given country code.

-- sample execution:

```
> pig -f task2b.pig -param MyPage=/user/hadoop/Proj1/data/MyPage.csv -param  
output=pig_task4b
```

Result:

File: [/user/hadoop/pig_task4b/part-r-00000](#)

Goto : [/user/hadoop/pig_task4b](#)

[Go back to dir listing](#)

[Advanced view/download options](#)

```
1      9860  
2      10026  
3      10058  
4      10117  
5      10110  
6      10083  
7      9919  
8      9815  
9      10042  
10     9970
```

For taskb, it used 19 seconds.

```
HadoopVersion  PigVersion      UserId  StartedAt      FinishedAt      Features  
1.2.1    0.13.0  hadoop  2017-09-22 00:12:47  2017-09-22 00:13:06  GROUP_BY  
  
Success!  
  
Job Stats (time in seconds):  
JobId   Maps   Reduces  MaxMapTime      MinMapTime      AvgMapTime      MedianMa  
pTime   MaxReduceTime  MinReduceTime  AvgReduceTime  MedianReducetime  A  
lias   Feature Outputs  
job_201709212003_0031  1       1       2       2       2       2       8       8  
8       8       data,nat_cit,nat_group  GROUP_BY,COMBINER  hdfs://localhost  
:8020/user/hadoop/pig_task4b,  
  
Input(s):  
Successfully read 100000 records (5599980 bytes) from: "/user/hadoop/Proj1/data/  
MyPage.csv"  
  
Output(s):  
Successfully stored 10 records (77 bytes) in: "hdfs://localhost:8020/user/hadoop/  
/pig_task4b"
```

task4c.)

Loads AccessLog data and groups data based on whatpage visited. It counts the number of times each unique id was viewed, sorts the counts in descending order, then takes the top 10.

-- sample execution:

```
>pig -f task4c.pig -param AccessLog=/pigdata/sampleAccessLog.csv -param output=sampleOutput
```

Result:

File: [/user/hadoop/pig_task4c/part-r-00000](#)

Goto : [/user/hadoop/pig_task4c](#)

[Go back to dir listing](#)
[Advanced view/download options](#)

```
59666 137  
37883 137  
24271 137  
38048 136  
62485 136  
63713 136  
38339 136  
33985 136  
54706 135  
53314 135
```

For this task, it used 187 seconds.

```
1.2.1 0.13.0 hadoop 2017-09-22 00:29:32      2017-09-22 00:32:39      GROUP_BY  
,ORDER_BY,LIMIT  
  
Success!  
  
Job Stats (time in seconds):  
JobId   Maps   Reduces  MaxMapTime    MinMapTime    AvgMapTime    MedianMa  
pTime   MaxReduceTime  MinReduceTime  AvgReduceTime  MedianReducetime  A  
lias   Feature Outputs  
job_201709212003_0036 9       1       27       12       23       23       92       9  
2       92      counted_log,data,grouped_log      GROUP_BY,COMBINER  
job_201709212003_0037 1       1       1       1       1       1       8       8  
8       8       ordered_log      SAMPLER  
job_201709212003_0038 1       1       1       1       1       1       8       8  
8       8       ordered_log      ORDER_BY,COMBINER  
job_201709212003_0039 1       1       1       1       1       1       8       8  
8       8       ordered_log      hdfs://localhost:8020/user/hadoop/pig_ta  
sk4c,  
  
Input(s):  
Successfully read 9655431 records (604015937 bytes) from: "/user/hadoop/Proj1/d  
ata/AccessLog.csv"  
  
Output(s):  
Successfully stored 10 records (100 bytes) in: "hdfs://localhost:8020/user/hadoo  
p/pig_task4c"
```

In Hadoop, PersonID is not ordered, so the results may be different.

Task4d.)

Loads in both mypage and friends data. Groups friend data by myfriend (ids that are the friend of someone else) and counts the number of instances. Joins the mypage and newly grouped datasets on id, and then takes the name and count columns to report.

-- sample execution:

```
> pig task4d.pig
```

Partial Result:

File: /user/hadoop/pig_task4d.out/part-r-00000

Goto : [/user/hadoop/pig_task4d.out](#) go

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

```
1 BggDsJuHTqkjw7 198
2 3rLrDcVHl1xVtCyj 196
3 2Tks3zdTT0kB5Gt 201
4 UXWIBVJ1xaBRbFrHYI 186
5 QllHyey2lKEqsJ 194
6 FdgRRDLMx7zon 193
7 dl90No95NMeRo 183
8 7byM05A0kRvyp4r4MLYD 206
9 6hv1jvU30Nk3iii 213
10 DjaB2pSuFFc0I 219
11 SgXYPXRRp4iBm 203
12 encvGrQ3hI 213
13 PloKWm0cq 216
14 Of7SGFjY4zgfGYIO 175
15 MynH2rFegwoS3Rqqhh0C 195
16 BgSjuWeVTY6EGq 193
17 xYRR7vPhfNOenC 208
18 VvyihVgWGkWVUP 208
19 ZpmODpdmh1e8QAbtx2q 199
```

For task d, it used 279 seconds.

```
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.2.1 0.13.0 hadoop 2017-09-22 01:22:39 2017-09-22 01:27:18 HASH_JOIN
N, GROUP_BY

Success!

Job Stats (time in seconds):
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMa
pTime MaxReduceTime MinReduceTime AvgReduceTime MedianReducetime A
alias Feature Outputs
job_201709212003_0047 20 2 25 15 23 23 219 2
19 219 219 countedFriends,friends,groupedFriends GROUP_BY,COMBINE
R
job_201709212003_0048 2 1 4 3 3 3 9 9
9 9 joinFL,myPage,selectedFL HASH_JOIN hdfs://localhost
:8020/user/hadoop/pig_task4d.out,
Input(s):
Successfully read 20000000 records (1312314832 bytes) from: "/user/hadoop/Proj1/
data/Friends.csv"
Successfully read 100000 records from: "/user/hadoop/Proj1/data/MyPage.csv"

Output(s):
Successfully stored 100000 records (2589094 bytes) in: "hdfs://localhost:8020/us
er/hadoop/pig_task4d.out"
```

task4e.)

Load accesslog and group by bywho id. Count the number of accesses by each id. Then find the unique pages and the number of accesses. Join the tables of unique and number of accesses.

-- sample execution:

```
> pig -f task4e.pig -param AccessLog=/user/hadoop/Proj1/data/AccessLog.csv -param  
output1=pig_task4e_1 -param output2=pig_task4e_2
```

Partial Result: pig_task4e_1: total, pig_task4e_2: distinct

File: /user/hadoop/pig_task4e_1/part-r-00000	File: /user/hadoop/pig_task4e_2/part-r-00000
Goto : /user/hadoop/pig_task4e_1 go	Goto : /user/hadoop/pig_task4e_2 go
Go back to dir listing	Go back to dir listing
Advanced view/download options	Advanced view/download options

View Next chunk	View Next chunk
<pre>1 88 2 78 3 99 4 98 5 101 6 107 7 103 8 100 9 106 10 93 11 94 12 96 13 94 14 91 15 105 16 87 17 88 18 80 19 86 20 86 21 105 22 100 23 97</pre>	<pre>1 88 2 78 3 99 4 98 5 101 6 107 7 103 8 100 9 106 10 93 11 94 12 96 13 94 14 91 15 105 16 87 17 88 18 80 19 86 20 86 21 105 22 99 23 97</pre>

For this task, it used 180 seconds.

```
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.2.1 0.13.0 hadoop 2017-09-22 01:46:51 2017-09-22 01:49:51 GROUP_BY

Success!

Job Stats (time in seconds):
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMa
pTime MaxReduceTime MinReduceTime AvgReduceTime MedianReducetime A
lias Feature Outputs
job_201709212003_0051 9 1 24 11 21 22 146 1
46 146 146 1-1,access_log,count_distinct_log, counted_log,grouped_lo
g,unique_page GROUP_BY,MULTI_QUERY hdfs://localhost:8020/user/hadoop/pig_ta
sk4e_1,hdfs://localhost:8020/user/hadoop/pig_task4e_2,

Input(s):
Successfully read 9655431 records (604015937 bytes) from: "/user/hadoop/Proj1/da
ta/AccessLog.csv"

Output(s):
Successfully stored 100000 records (926504 bytes) in: "hdfs://localhost:8020/use
r/hadoop/pig_task4e_1"
Successfully stored 100000 records (926318 bytes) in: "hdfs://localhost:8020/use
r/hadoop/pig_task4e_2"
```

task4f.)

Due to people can check their friends' page many time, so first find unique accesslog of ByWho and WhatPage, and then, find the difference between friends dataset and this unique log dataset.

-- sample execution:

```
> pig -f task4f.pig -param Friends=/user/hadoop/Proj1/data/Friends.csv -param AccessLog=/user/hadoop/Proj1/data/AccessLog.csv -param output=pig_task4f
```

Partial Result: The first element of tuple is the person id, the second element is the id of friend

Goto : [/user/hadoop/pig_task4f](#) go

[Go back to dir listing](#)
[Advanced view/download options](#)

[View Next chunk](#)

```
{(2,99771), (2,51870), (2,66223), (2,95269), (2,15643), (2,96949), (2,74894), (2,55095), (2,1755), (2,75838), (2,36709), (2,31901), (2,19915), (2,73055), (2,77349), (2,40030), (2,29682), (2,4485), (2,59608), (2,65450), (2,20490), (2,2035), (2,96449), (2,8043), (2,33177), (2,71930), (2,76306), (2,2118), (2,27001), (2,9402), (2,81250), (2,87567), (2,8854), (2,48715), (2,95340), (2,17242), (2,51408), (2,16695), (2,6508), (2,49737), (2,82738), (2,57870), (2,77364), (2,70071), (2,62772), (2,57362), (2,58461), (2,30307), (2,99269), (2,88806), (2,48181), (2,97479), (2,35171), (2,58936), (2,58506), (2,61905), (2,31763), (2,63373), (2,26952), (2,59468), (2,96838), (2,20594), (2,78130), (2,26031), (2,80321), (2,62844), (2,21687), (2,74313), (2,44433), (2,5988), (2,47826), (2,78151), (2,76488), (2,20303), (2,46162), (2,66572), (2,36780), (2,84284), (2,78115), (2,86712), (2,27434), (2,71426), (2,93538), (2,52725), (2,68278), (2,50832), (2,61005), (2,74833), (2,69257), (2,88133), (2,48216), (2,7137), (2,77514), (2,42333), (2,50493), (2,66994), (2,63711), (2,66461), (2,24045), (2,97690), (2,47466), (2,13479), (2,34535), (2,54837), (2,66469), (2,11062), (2,69770), (2,49431), (2,25627), (2,70309), (2,82435), (2,12042), (2,415), (2,10102), (2,60824), (2,29870), (2,44533), (2,24545), (2,15350), (2,48399), (2,49453), (2,91745), (2,1415), (2,63641), (2,84695), (2,89996), (2,6251), (2,15937), (2,91191), (2,36375), (2,20826), (2,75746), (2,11599), (2,63038), (2,99498), (2,43524), (2,71739), (2,21340), (2,40198), (2,22290), (2,59214), (2,71753), (2,52146), (2,15409), (2,5263), (2,2892), (2,1434), (2,69913), (2,66021), (2,47909), (2,7701), (2,27649), (2,5691), (2,65790), (2,76773), (2,248), (2,30108), (2,51448), (2,11652), (2,83126), (2,40417), (2,97542), (2,24407), (2,16533), (2,61191), (2,59754), (2,64523), (2,49066), (2,57831), (2,22483), (2,88306), (2,9020), (2,29092), (2,68047), (2,38179), (2,41489), (2,36506), (2,85538), (2,60190), (2,58831), (2,72387), (2,59648), (2,97066), (2,17565), (2,841), (2,29591), (2,76191), (2,56880), (2,34305), (2,52505), (2,14456)}
```

```
{(4,42319), (4,53946), (4,56744), (4,88827), (4,95695), (4,37440), (4,59536), (4,43821), (4,4445), (4,6386), (4,76332), (4,29681), (4,11730), (4,39410), (4,46137), (4,80139), (4,22663), (4,18187), (4,67194), (4,66110), (4,89776), (4,37486), (4,38842), (4,72413), (4,28677), (4,16600), (4,42224), (4,6905), (4,8895), (4,47556), (4,11817), (4,19360), (4,95275), (4,89651), (4,15039), (4,96827), (4,59469), (4,686), (4,43747), (4,60569), (4,52794), (4,50635), (4,9843), (4,57809), (4,63816), (4,5023), (4,25280), (4,16673), (4,59983), (4,23191), (4,92929), (4,70991), (4,77399), (4,62338), (4,42182), (4,11332), (4,39011), (4,52215), (4,3759), (4,23597), (4,13068), (4,73746), (4,74689), (4,48222), (4,63314), (4,19171), (4,18651), (4,54649), (4,50819), (4,14718), (4,47780), (4,99998), (4,16222), (4,90526), (4,37337), (4,91008), (4,30724), (4,79058), (4,56573), (4,82756), (4,73621), (4,89639), (4,96804), (4,47230), (4,40600), (4,56069), (4,73669), (4,59860), (4,19642), (4,9215), (4,65288), (4,33996), (4,55597), (4,12524), (4,16759), (4,28773), (4,41147), (4,95381), (4,51699), (4,69172), (4,19668), (4,13034), (4,86149), (4,38686), (4,67733), (4,82335), (4,56522), (4,83287), (4,81836), (4,16296), (4,41), (4,95353), (4,36888), (4,2202), (4,35247), (4,63647), (4,29910), (4,99451), (4,34747), (4,95942), (4,23363), (4,9113), (4,1954), (4,20115), (4,42026), (4,28950), (4,39140), (4,7180), (4,56436), (4,54831), (4,23391), (4,54280), (4,44013), (4,37175), (4,61973), (4,95430), (4,85166), (4,2275), (4,59085), (4,5261), (4,2768), (4,53763), (4,82479), (4,75680), (4,81396), (4,5698), (4,23430), (4,96006), (4,77599), (4,74668), (4,15380), (4,7605), (4,54233), (4,90581), (4,31578), (4,7121), (4,32129), (4,2806), (4,69318), (4,1867), (4,27213), (4,92732), (4,80908), (4,13987), (4,76476), (4,85091), (4,66923), (4,43448), (4,88782), (4,31006), (4,60625),
```

For this task, it used 451 seconds.

HadoopVersion	PigVersion	UserId	StartedAt	FinishedAt	Features
1.2.1	0.13.0	hadoop	2017-09-22 02:06:47	2017-09-22 02:14:18	COGROUP, DISTINCT
Success!					
Job Stats (time in seconds):					
JobId	Maps	Reduces	MaxMapTime	MinMapTime	AvgMapTime
pTime	MaxReduceTime	MinReduceTime	AvgReduceTime	MedianReducetime	A
alias	Feature	Outputs			
job_201709212003_0054	9	1	28	11	21
18	118	118	accessLog,sAccessLog	DISTINCT	
job_201709212003_0055	22	2	42	9	19
70	270	270	cogroup_data,friends,sFriends,sub_data	COGROUP	hdfs://localhost:8020/user/hadoop/pig_task4f,
Input(s):					
Successfully read 9655431 records (604015937 bytes) from: "/user/hadoop/Proj1/data/AccessLog.csv"					
Successfully read 20000000 records from: "/user/hadoop/Proj1/data/Friends.csv"					
Output(s):					
Successfully stored 100000 records (275216211 bytes) in: "hdfs://localhost:8020/user/hadoop/pig_task4f"					

Task4g.)

Load in the access log data. Group the logs by who id and compute the max accesstime for each id (this is the last time the person accessed facebook). Filter all those that have less than some time unit, ie havent been on facebook past that date.

-- sample execution:

```
> pig -f task4g.pig -param AccessLog=/user/hadoop/Proj1/data/AccessLog.csv -param  
time_unit=900000 -param output=pig_task4g
```

Partial Result:

Goto : [/user/hadoop/pig_task4g](#)

[Go back to dir listing](#)
[Advanced view/download options](#)

```
7078 892232  
17132 882939  
28202 882067  
30016 896912  
40552 885653  
41101 891638  
42486 897203  
48304 872763  
69346 894007  
73861 894653  
89929 897436  
94441 898422  
95032 892656  
96832 891048
```

For this task, it used 120 seconds.

```
HadoopVersion PigVersion UserId StartedAt FinishedAt Features  
1.2.1 0.13.0 hadoop 2017-09-22 11:01:23 2017-09-22 11:03:23 GROUP_BY  
,FILTER
```

Success!

Job Stats (time in seconds):

```
JobId Maps Reduces MaxMapTime MinMapTime AvgMapTime MedianMa  
pTime MaxReduceTime MinReduceTime AvgReduceTime MedianReducetime A  
lias Feature Outputs  
job_201709221039_0004 9 1 23 11 21 22 86 8  
6 86 access_log,filtered_log,grouped_log,max_log GROUP_BY  
,REDUCE hdfs://localhost:8020/user/hadoop/pig_task4g,
```

Input(s):

```
Successfully read 9655431 records (604015937 bytes) from: "/user/hadoop/Proj1/da  
ta/AccessLog.csv"
```

Output(s):

```
Successfully stored 14 records (181 bytes) in: "hdfs://localhost:8020/user/hadoo  
p/pig_task4g"
```

task4h.)

Load in friends data. Group by id and count the number of friends each person has. Calculated the average by grouping all people into one group, count the total number of friends, and divide by total ids. Then filter all people with more than the average.

-- sample execution:

```
> pig -f task4h.pig -param Friends=/user/hadoop/Proj1/data/Friends.csv -param output=pig_task4h
```

Partial Result:

Goto : /user/hadoop/pig_task4h

[Go back to dir listing](#)

[Advanced view/download options](#)

[View Next chunk](#)

```
4    214
12   201
18   228
20   202
24   209
26   217
28   206
30   205
36   201
40   222
44   207
46   200
58   200
60   233
62   217
70   205
72   206
74   211
78   213
80   209
86   201
88   205
```

For this task, it used 258 seconds.

```
HadoopVersion  PigVersion      UserId  StartedAt      FinishedAt      Features
1.2.1  0.13.0  hadoop  2017-09-22 02:50:16  2017-09-22 02:54:34  GROUP_BY
,FILTER

Success!

Job Stats (time in seconds):
JobId  Maps     Reduces  MaxMapTime      MinMapTime      AvgMapTime      MedianMa
pTime  MaxReduceTime  MinReduceTime  AvgReduceTime  MedianReducetime  A
lias  Feature Outputs
job_201709212003_0059  20       2        22           12           20           21           194          1
94       194       194       counted_friends,data,grouped_friends  GROUP_BY,COMBINE
R
job_201709212003_0060  1        1        1            1            1            1            8            8
8         8         grouped_avg,grouped_counted  GROUP_BY,COMBINER
job_201709212003_0061  1        0        2            2            2            2            0            0
0         0         filtered_friends  MAP_ONLY        hdfs://localhost:8020/us
er/hadoop/pig_task4h,

Input(s):
Successfully read 20000000 records (1312314832 bytes) from: "/user/hadoop/Proj1/
data/Friends.csv"

Output(s):
Successfully stored 50903 records (503327 bytes) in: "hdfs://localhost:8020/user
```

	Mapreduce without combiner	Mapreduce with combiner	Pig
a	0.21s	NA	14s
b	0.69s	0.6s	19s
c	38.76s	NA	187s
d	251.71s	NA	279s
e	51.46s	NA	180s
f	178.08s	NA	451s
g	49.47s	NA	120s
h	103.12s	NA	258s

Based on this table, for Hadoop, not every task could use combiner, but when can use it, the one with combiner will be quicker than the one without. For Pig and Hadoop, Hadoop has a better performance in time than Pig does, especially for small tasks. And for Hadoop, you can write any kind of function as you can, but for Pig, you only can use the built-in functions. Hadoop is more flexible than Pig. For Pig, it is much easier to learn than Hadoop. To choose Hadoop or Pig depends on people's background and the purpose of their tasks.