MIDS-W261-2016-HW-Week03-Lane

June 5, 2016

1 DATASCI W261: Machine Learning at Scale

1.1 Assignment Week 3

Jackson Lane (jelane@berkeley.edu) W261-3

1.2 HW3.0.

How do you merge two sorted lists/arrays of records of the form [key, value]? Where is this used in Hadoop MapReduce? [Hint within the shuffle] If you're starting out with two sorted lists, it's pretty easy to merge them. First initalize a new empty list to hold all the elements from your two input lists. Then out of all the elements in the two input lists, remove the element with the smallest key and put it at the begining of the new list. Then out of the remaining elements from the two input lists, take the element with the smallest and put it in the second position in the new list. Repeat this step until you have no more elements left in either input list. You should now have a sorted list.

Because the two input lists are already sorted, you only need to make one comparison per step to find the smallest key. As such, the algorithm can run in O(n+m) time, where n and m are the size of the first and second input lists respectively.

The interactive HTML below does a better job of explaining the process than I do.

```
In [ ]: %%HTML
        <div style="width:700px; height:400px; overflow:hidden;  position:relative; margin: 0px auto; ba
        <iframe src="http://cs.armstrong.edu/liang/animation/web/MergeList.html" scrolling="no" style="
        </div>
```

In Hadoop, this type of sort is applied after the combining phase in the Hadoop shuffle. If there is more than 1 mapper, then Hadoop will merge the outputs of the two mappers and/or combiners together on the reducer before executing the actual reducer code. Hadoop can do this, because it assumes that both of these outputs were already sorted during the shuffle. This type of sorting also happens during the Hadoop shuffle multiple times in a process called "merge-sort" that takes two unsorted lists and returns a sorted list. The merge-sort process happens during the shuffle phasse, after partitioning but before combining. It works by breaking down the lists into single elements and then merging these single elements back into larger sorted lists.

What is a combiner function in the context of Hadoop? Give an example where it can be used and justify why it should be used in the context of this problem. A combiner function is like a reducer run on the mapper side. It happens during the shuffle phase and runs on elements with the same key. It's used to reduce the volumne of data sent from the mappers to the reducers over the network. A good example is in the generic word count example. Basically, instead of sending over 50 instances of 1 count of the same word for the reducer to count, you can use a combiner to aggregate those 50 instances into just 1 instance with a count of 50. In java the combiner may be applied once per unique key emitted by the mappers. However, in Hadoop Streaming, it appears that combiner can sometimes run on multiple keys at a time. In both cases however, Hadoop does not guarantee that it will run the combiner or the amount of times it will run the combiner. It could run 0, 1, or many times.

What is the Hadoop shuffle? The Hadoop Shuffle is the process Hadoop uses to transfer the outputs from the mappers to reducers. The shuffle consists of three parts: Partitioning, sorting, and combining. The first two phases always happen in that order in each shuffle phase, but the third phase may or may not happen, hence why Hadoop does not guarantee it will run combiners. Partitioning divides up the mapper outputs among the reducer tasks. Typically this is done using a hash, but you can specify other types of partitioners as well in the job configuration. Once partitioning has completed, the sorting phase sorts each partition using merge-sort. Again, this behavior is configurable in the job configuration. Then the combiner phase may execute a combiner funciton on the sorted partitions. Note that the combiner cannot change which reducer a record gets sent to, as that is determined in the partitioning pase. But a combiner change the order in which the reducer recieves the records by printing the records in a different order than received from the sort phase.

1.3 HW3.1

Use Counters to do EDA (exploratory data analysis and to monitor progress) Counters are lightweight objects in Hadoop that allow you to keep track of system progress in both the map and reduce stages of processing. By default, Hadoop defines a number of standard counters in "groups"; these show up in the jobtracker webapp, giving you information such as "Map input records", "Map output records", etc.

While processing information/data using MapReduce job, it is a challenge to monitor the progress of parallel threads running across nodes of distributed clusters. Moreover, it is also complicated to distinguish between the data that has been processed and the data which is yet to be processed. The MapReduce Framework offers a provision of user-defined Counters, which can be effectively utilized to monitor the progress of data across nodes of distributed clusters.

Use the Consumer Complaints Dataset provide here to complete this question:

`https://www.dropbox.com/s/vbalm3yva2rr86m/Consumer_Complaints.csv?dl=0`

The consumer complaints dataset consists of diverse consumer complaints, which have been reported across the United States regarding various types of loans. The dataset consists of records of the form:

Complaint ID,Product,Sub-product,Issue,Sub-issue,State,ZIP code,Submitted via,Date received,Date sent to company,Company,Company response,Timely response?,Consumer disputed?

User-defined Counters

Now, let's use Hadoop Counters to identify the number of complaints pertaining to debt collection, mortgage and other categories (all other categories get lumped into this one) in the consumer complaints dataset. Basically produce the distribution of the Product column in this dataset using counters (limited to 3 counters here).

Hadoop offers Job Tracker, an UI tool to determine the status and statistics of all jobs. Using the job tracker UI, developers can view the Counters that have been created. Screenshot your job tracker UI as your job completes and include it here. Make sure that your user defined counters are visible.

```
In [ ]: %%writefile mapper.py
        #!/usr/bin/python
        # mapper.py
        # Author:Jackson Lane
        # Description: mapper code for HW3.1

        import sys

        for line in sys.stdin:
            fields = line.split(",")
            product = fields[1]
            #Group all other product fields into just "other"
            if product != "Debt collection" and product != "Mortgage": product = "Other"
            #Increment counter by 1
            sys.stderr.write("reporter:counter:3.1,"+product+",1\n")
            print product,",", 1
```

In [ ]:
```
%%writefile reducer.py
#!/usr/bin/python
# reducer.py
# Author: Jackson Lane
# Description: reducer code for HW3.1

from __future__ import print_function
from operator import itemgetter
import sys

word = ""
count = 0

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    newword, newcount = line.split(",")
    newcount = 1
    if (newword == word): count += newcount
    else:
        # We have finished with all instances of the current word.
        # Print total count and move on to next word
        if (count > 0): print (word, count ,sep=',')
        word = newword
        count = newcount
if (count > 0): print (word, count , sep=',')
```

Run the mapreduce job

In [ ]:
```
!hdfs dfs -rm -r results/3.1
!hdfs dfs -put Consumer_Complaints.csv
!hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
-D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparat
-D mapreduce.partition.keycomparator.options=-n \
-D mapreduce.output.key.field.separator="," \
-file mapper.py \
-file reducer.py \
-mapper "mapper.py" \
-reducer "reducer.py" \
-input Consumer_Complaints.csv \
-output results/3.1
```

In [ ]:
```
!hdfs dfs -cat results/3.1/part-00000
```

In [ ]:
```
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "https://dl.dropboxusercontent.com/u/43045211/stats/HW3/HW3.1.png.PNG")
```

1.4 HW 3.2

Analyze the performance of your Mappers, Combiners and Reducers using Counters

1.4.1 HW 3.2.0.1

Perform a word count analysis of this single record dataset using a Mapper and Reducer based WordCount (i.e., no combiners are used here) using user defined Counters to count up how many time the mapper and

reducer are called. What is the value of your user defined Mapper Counter, and Reducer Counter after completing this word count job. The answer should be 1 and 4 respectively. Please explain.

In [ ]:
```
%%writefile mapper.py
#!/usr/bin/python
# mapper.py
# Author:Jackson Lane
# Description: mapper code for HW3.2.0.1

import sys
sys.stderr.write("reporter:counter:3.2,MapperCount,1\n")

for line in sys.stdin:
    line = line.split()
    for word in line:
        #Emit each word with a count of 1
        print word,",",1
```

In [ ]:
```
%%writefile reducer.py
#!/usr/bin/python
# reducer.py
# Author: Jackson Lane
# Description: reducer code for 3.2.0.1
# Same as reducer code for 3.1, but with a counter
from __future__ import print_function
from operator import itemgetter
import sys

word = ""
count = 0
sys.stderr.write("reporter:counter:3.2,ReducerCount,1\n")

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    newword, newcount = line.split(",")
    newcount = int(newcount)
    if (newword == word): count += newcount
    else:
        # We have finished with all instances of the current word.
        # Print total count and move on to next word
        if (count > 0): print (word, count, sep=',')
        word = newword
        count = newcount
if (count > 0): print (word,  count, sep=',')
```

make the input file

In [ ]:
```
%%writefile ffqlfbq.txt
foo foo quux labs foo bar quux
```

Run the mapreduce job

In [ ]:
```
!hdfs dfs -rm -r results/3.2
!hdfs dfs -put -p -f ffqlfbq.txt
```

```
!hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
-D mapred.map.tasks=1 \
-D mapred.reduce.tasks=4 \
-file mapper.py \
-file reducer.py \
-mapper "mapper.py" \
-reducer "reducer.py" \
-input ffqlfbq.txt \
-output results/3.2
```

```
In [ ]: !hdfs dfs -cat results/3.2/part-00000
```

The mapper and reducer counters are equal to the number of mapper and reducer jobs respectively. The default seems to be 2 mapper jobs and 1 reducer job. However, I can explicitly set the mapper jobs to 1 and the reducer jobs to 4 in the run command. Perhaps in java though, the reducer tasks will run on each unique key by default, but this behavior does not appear to be the case in Hadoop Streaming.

1.4.2 HW 3.2.0.2

Please use mulitple mappers and reducers for these jobs (at least 2 mappers and 2 reducers). Perform a word count analysis of the Issue column of the Consumer Complaints Dataset using a Mapper and Reducer based WordCount (i.e., no combiners used anywhere) using user defined Counters to count up how many time the mapper and reducer are called. What is the value of your user defined Mapper Counter, and Reducer Counter after completing your word count job.

```
In [20]: %%writefile mapper.py
         #!/usr/bin/python
         # mapper.py
         # Author:Jackson Lane
         # Description: mapper code for 3.2.0.2 and 3.2.0.3
         from __future__ import print_function

         import sys,re
         sys.stderr.write("reporter:counter:3.2,MapperCount,1\n")

         for line in sys.stdin:
             fields = line.split(",")
             #Get the issue field
             issue = fields[3]
             #split the issue field into individual words
             words = re.findall("[\w']+",issue)
             for word in words:
                 word = word.lower()
                 # Even though we have two reducers, we still want to print out
                 # an alphabetically sorted list of word counts.  So we make sure
                 # to send every word that begins with a-l to the first partition
                 # and everything else to the second partition
                 partitionkey = int(word > "m")
                 #Emit each word with a count of 1
                 print (partitionkey,word,1,sep=",")
```

```
Overwriting mapper.py
```

```
In [21]: %%writefile reducer.py
         #!/usr/bin/python
         # reducer.py
```

```
# Author: Jackson Lane
# Description: reducer code for 3.2.0.2, 3.2.0.3
# Same as reducer code for 3.2.0.1, but adjusting for the extra partitionkey field
from __future__ import print_function
from operator import itemgetter
import sys

word = ""
count = 0
sys.stderr.write("reporter:counter:3.2,ReducerCount,1\n")

for line in sys.stdin:
    sys.stderr.write(line)

    # remove leading and trailing whitespace
    line = line.strip()
    _,newword, newcount = line.split(",")
    newcount = int(newcount)
    if (newword == word): count += newcount
    else:
        # We have finished with all instances of the current word.
        # Print total count and move on to next word
        if (count > 0): print (word,count,sep=',')
        word = newword
        count = newcount
if (count > 0): print (word,count,sep=',')
```

```
Overwriting reducer.py
```

Run the map-reduce job

```
In [22]: !hdfs dfs -rm -r results/3.2
         !hdfs dfs -put -p -f Consumer_Complaints.csv
         !hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
         -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCompara
         -D mapreduce.partition.keypartitioner.options="-k1,1n" \
         -D mapreduce.partition.keycomparator.options="-k2,2" \
         -D mapreduce.output.key.field.separator=, \
         -D stream.map.output.field.separator=, \
         -D stream.reduce.output.field.separator=, \
         -D stream.map.input.field.separator=, \
         -D stream.reduce.input.field.separator=, \
         -D map.output.key.field.separator=, \
         -D stream.num.map.output.key.fields=2 \
         -D mapred.map.tasks=2 \
         -D mapred.reduce.tasks=2 \
         -file mapper.py \
         -file reducer.py \
         -mapper "mapper.py" \
         -reducer "reducer.py" \
         -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \
         -input Consumer_Complaints.csv \
         -output results/3.2
```

```
16/06/05 11:02:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 11:02:37 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minut
```

```
Deleted results/3.2
16/06/05 11:02:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 11:02:41 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/06/05 11:02:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-unjar8007677370073082870/] [] /tmp/streamjob771309811
16/06/05 11:02:42 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:02:42 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:02:42 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 11:02:42 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 11:02:42 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapred
16/06/05 11:02:42 INFO Configuration.deprecation: map.output.key.field.separator is deprecated. Instead
16/06/05 11:02:42 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce
16/06/05 11:02:43 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0187
16/06/05 11:02:43 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0187
16/06/05 11:02:43 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicati
16/06/05 11:02:43 INFO mapreduce.Job: Running job: job_1464324416493_0187
16/06/05 11:02:50 INFO mapreduce.Job: Job job_1464324416493_0187 running in uber mode : false
16/06/05 11:02:50 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 11:02:59 INFO mapreduce.Job:  map 50% reduce 0%
16/06/05 11:03:00 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 11:03:09 INFO mapreduce.Job:  map 100% reduce 50%
16/06/05 11:03:10 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 11:03:12 INFO mapreduce.Job: Job job_1464324416493_0187 completed successfully
16/06/05 11:03:13 INFO mapreduce.Job: Counters: 53
        File System Counters
                FILE: Number of bytes read=13194441
                FILE: Number of bytes written=26879192
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50910610
                HDFS: Number of bytes written=2091
                HDFS: Number of read operations=12
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Job Counters
                Killed reduce tasks=1
                Launched map tasks=2
                Launched reduce tasks=2
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=57940
                Total time spent by all reduces in occupied slots (ms)=135736
                Total time spent by all map tasks (ms)=14485
                Total time spent by all reduce tasks (ms)=16967
                Total vcore-milliseconds taken by all map tasks=14485
                Total vcore-milliseconds taken by all reduce tasks=16967
                Total megabyte-milliseconds taken by all map tasks=59330560
                Total megabyte-milliseconds taken by all reduce tasks=138993664
        Map-Reduce Framework
                Map input records=312912
                Map output records=980482
                Map output bytes=11233465
                Map output materialized bytes=13194453
```

```
                Input split bytes=202
                Combine input records=0
                Combine output records=0
                Reduce input groups=169
                Reduce shuffle bytes=13194453
                Reduce input records=980482
                Reduce output records=169
                Spilled Records=1960964
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=343
                CPU time spent (ms)=20990
                Physical memory (bytes) snapshot=1130864640
                Virtual memory (bytes) snapshot=27566129152
                Total committed heap usage (bytes)=1036517376
        3.2
                MapperCount=2
                ReducerCount=2
        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=50910408
        File Output Format Counters
                Bytes Written=2091
16/06/05 11:03:13 INFO streaming.StreamJob: Output directory: results/3.2
```

From the output, you can see that the mapper and reducer counts are both 2.

```
In [23]: !hdfs dfs -cat results/3.2/part-0000*
```

```
16/06/05 11:03:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
a        3503
account        20681
acct        163
action        2505
advance        240
advertising        1193
amount        98
amt        71
an        2505
and        16448
application        8868
applied        139
apply        118
apr        3431
arbitration        168
are        3821
atm        2422
attempts        11848
```

```
available        274
balance        597
bank        202
bankruptcy        222
being        5663
billing        8158
by        5663
can’t        1999
cancelling        2795
card        4405
cash        240
caused        5663
changes        350
charged        976
charges        131
checks        75
closing        2795
club        12545
collect        11848
collection        1907
communication        6920
company’s        4858
cont’d        11848
contact        3053
convenience        75
costs        4350
credit        55251
credited        92
customer        2734
day        71
dealing        1944
debit        2422
debt        19309
decision        2774
decrease        1149
delay        243
delinquent        1061
deposits        10555
determination        1490
did        139
didn’t        925
disclosure        5214
disclosures        64
dispute        904
disputes        6938
embezzlement        3276
expect        807
false        2508
fee        3198
fees        807
for        929
forbearance        350
fraud        3842
funds        5663
```

```
get         4357
getting          291
health          12545
i         925
identity          4729
illegal          2505
improper          4309
incorrect           29133
increase          1149
info         2896
information           29069
interest          4238
investigation          4858
issuance          640
issue          1098
issues          538
late          1797
lease          6337
lender          2165
line          1732
loan          119630
low          5663
making          3226
managing          5006
marketing           1193
missing          64
modification           70487
money          413
monitoring           1453
my          10731
not          12353
of          10885
on          29069
opening           16205
or          22533
other           7886
out          1242
overlimit           127
owed          11848
pay          3821
payment          92
payments          3226
payoff          1155
plans          350
practices           1003
privacy          240
problems          9484
process          5505
processing           243
promised          274
protection           4139
rate          3431
receive          139
received          216
```

```
receiving        3226
relations        1367
repay        1647
repaying        3844
report        34903
reporting        6559
representation        2508
rewards        1002
sale        139
scam        566
score        4357
service        1518
servicer        1944
servicing        36767
settlement        4350
sharing        2832
shopping        672
statement        1220
statements        2508
stop        131
tactics        6920
taking        3747
terms        350
the        6248
theft        3276
threatening        2505
to        8401
transaction        1485
transfer        597
unable        8178
underwriting        2774
unsolicited        640
use        1477
using        2422
verification        5214
was        274
when        4095
with        1944
withdrawals        10555
workout        350
wrong        169
you        3821
your        3844
```

1.4.3 HW 3.2.0.3

Perform a word count analysis of the Issue column of the Consumer Complaints Dataset using a Mapper, Reducer, and standalone combiner (i.e., not an in-memory combiner) based WordCount using user defined Counters to count up how many time the mapper, combiner, reducer are called. What is the value of your user defined Mapper Counter, and Reducer Counter after completing your word count job?

In [13]:
```
# Use same mapper as in previous problem
```

```
Overwriting mapper.py
```

In [24]:
```
%%writefile combiner.py
#!/usr/bin/python
# combiner.py
# Author: Jackson Lane
# Description: combiner code for 3.2.0.3
# Similar to reducer code from 3.2.0.2, except that the combiner
# also prints the partition key and has a different counter
from __future__ import print_function
import sys

word = ""
count = 0
sys.stderr.write("reporter:counter:3.2,CombinerCount,1\n")

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    partitionkey,newword, newcount = line.split(",")
    newcount = int(newcount)
    if (newword == word): count += newcount
    else:
        # We have finished with all instances of the current word.
        # Print total count and move on to next word
        if (count > 0): print (partitionkey,word, count, sep=',')
        word = newword
        count = newcount
if (count > 0): print (partitionkey,word,  count, sep=',')
```

```
Overwriting combiner.py
```

In [25]:
```
# Use same reducer as in previous problem
```

Run map reduce job

In [26]:
```
!hdfs dfs -rm -r results/3.2
!hdfs dfs -put -p -f Consumer_Complaints.csv
!hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
-D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCompara
-D mapreduce.partition.keypartitioner.options="-k1,1n" \
-D mapreduce.partition.keycomparator.options="-k2,2" \
-D mapreduce.output.key.field.separator=, \
-D mapreduce.map.output.key.field.separator=, \
-D stream.map.output.field.separator=, \
-D stream.reduce.output.field.separator=, \
-D stream.map.input.field.separator=, \
-D stream.reduce.input.field.separator=, \
-D stream.num.map.output.key.fields=2 \
-file mapper.py \
-file reducer.py \
-file combiner.py \
-mapper "mapper.py" \
-combiner "combiner.py" \
-reducer "reducer.py" \
-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \
-input Consumer_Complaints.csv \
-output results/3.2
```

```
16/06/05 11:04:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 11:04:29 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minute
Deleted results/3.2
16/06/05 11:04:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 11:04:33 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/06/05 11:04:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, reducer.py, combiner.py, /tmp/hadoop-unjar5076983616027755871/] [] /tmp/strea
16/06/05 11:04:34 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:04:34 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:04:35 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 11:04:35 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 11:04:36 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0188
16/06/05 11:04:36 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0188
16/06/05 11:04:36 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicati
16/06/05 11:04:36 INFO mapreduce.Job: Running job: job_1464324416493_0188
16/06/05 11:04:42 INFO mapreduce.Job: Job job_1464324416493_0188 running in uber mode : false
16/06/05 11:04:42 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 11:04:52 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 11:04:57 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 11:04:57 INFO mapreduce.Job: Job job_1464324416493_0188 completed successfully
16/06/05 11:04:57 INFO mapreduce.Job: Counters: 53
        File System Counters
                FILE: Number of bytes read=5064
                FILE: Number of bytes written=379727
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50910610
                HDFS: Number of bytes written=3739
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=64184
                Total time spent by all reduces in occupied slots (ms)=25184
                Total time spent by all map tasks (ms)=16046
                Total time spent by all reduce tasks (ms)=3148
                Total vcore-milliseconds taken by all map tasks=16046
                Total vcore-milliseconds taken by all reduce tasks=3148
                Total megabyte-milliseconds taken by all map tasks=65724416
                Total megabyte-milliseconds taken by all reduce tasks=25788416
        Map-Reduce Framework
                Map input records=312912
                Map output records=980482
                Map output bytes=11233465
                Map output materialized bytes=5070
                Input split bytes=202
                Combine input records=980482
                Combine output records=313
                Reduce input groups=28
```

```
                Reduce shuffle bytes=5070
                Reduce input records=313
                Reduce output records=307
                Spilled Records=626
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=224
                CPU time spent (ms)=8220
                Physical memory (bytes) snapshot=859910144
                Virtual memory (bytes) snapshot=19001569280
                Total committed heap usage (bytes)=740294656
        3.2
                CombinerCount=2
                MapperCount=2
                ReducerCount=1
        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=50910408
        File Output Format Counters
                Bytes Written=3739
16/06/05 11:04:57 INFO streaming.StreamJob: Output directory: results/3.2
```

```
In [28]: !hdfs dfs -cat results/3.2/part-0000*
```

```
16/06/05 11:05:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
a        3503
account        20681
acct        163
action        2505
advance        240
advertising        1193
amount        98
an        229
amt        71
and        10227
an        2276
application        5962
and        6221
apr        2557
application        2906
arbitration        92
applied        139
are        2510
apply        118
atm        1366
apr        874
attempts        1372
arbitration        76
```

```
available        64
are        1311
balance        400
atm        1056
bankruptcy        143
attempts        10476
being        3743
available        210
billing        5289
balance        197
by        3743
bank        202
cancelling        1822
bankruptcy        79
card        2673
being        1920
cash        167
billing        2869
caused        3743
by        1920
changes        235
can't        1999
charged        17
cancelling        973
checks        49
card        1732
closing        1822
cash        73
club        1783
caused        1920
collect        1372
changes        115
collection        1907
charged        959
communication        795
charges        131
company's        1913
checks        26
cont'd        1372
closing        973
contact        282
club        10762
convenience        49
collect        10476
costs        2811
communication        6125
credit        21686
company's        2945
customer        1532
cont'd        10476
debit        1366
contact        2771
debt        3920
convenience        26
```

```
decision        1823
costs        1539
decrease        826
credit        33565
delay        170
credited        92
delinquent        460
customer        1202
deposits        6488
day        71
determination        960
dealing        1944
disclosure        697
debit        1056
disclosures        14
debt        15389
dispute        904
decision        951
disputes        4528
decrease        323
embezzlement        1929
delay        73
false        275
delinquent        601
fee        2080
deposits        4067
for        287
determination        530
forbearance        256
did        139
fraud        2052
didn't        925
funds        3743
disclosure        4517
get        1489
disclosures        50
getting        193
disputes        2410
health        1783
embezzlement        1347
identity        2390
expect        807
illegal        229
false        2233
improper        940
fee        1118
incorrect        8711
fees        807
increase        826
for        642
info        296
forbearance        94
information        8697
fraud        1790
```

```
interest        2557
funds        1920
investigation        1913
get        2868
issuance        360
getting        98
issue        636
health        10762
issues        139
i        925
late        1142
identity        2339
lease        2909
illegal        2276
line        1188
improper        3369
loan        76245
incorrect        20422
low        3743
increase        323
making        1964
info        2600
managing        2485
information        20372
marketing        693
interest        1681
missing        14
investigation        2945
modification        48670
issuance        280
money        64
issue        462
monitoring        461
issues        399
my        4401
late        655
not        1436
lease        3428
of        2438
lender        2165
on        8697
line        544
opening        9567
loan        43385
or        8004
low        1920
other        4654
making        1262
out        499
managing        2521
overlimit        84
marketing        500
owed        1372
missing        50
```

```
pay        2510
modification        21817
payments        1964
money        349
payoff        802
monitoring        992
plans        256
my        6330
practices        1003
not        10917
privacy        141
of        8447
problems        6253
on        20372
process        3613
opening        6638
processing        170
or        14529
promised        64
other        3232
protection        2355
out        743
rate        2557
overlimit        43
received        17
owed        10476
receiving        1964
pay        1311
relations        766
payment        92
repaying        3409
payments        1262
report        10844
payoff        353
reporting        3614
plans        94
representation        275
privacy        99
rewards        671
problems        3231
sale        79
process        1892
scam        123
processing        73
score        1489
promised        210
service        794
protection        1784
servicing        21064
rate        874
settlement        2811
receive        139
sharing        282
received        199
```

```
shopping        287
receiving        1262
statement        761
relations        601
statements        275
repay        1647
tactics        795
repaying        435
taking        728
report        24059
terms        235
reporting        2945
the        2984
representation        2233
theft        1929
rewards        331
threatening        229
sale        60
to        3999
scam        443
transaction        747
score        2868
transfer        400
service        724
unable        3999
servicer        1944
underwriting        1823
servicing        15703
unsolicited        360
settlement        1539
use        658
sharing        2550
using        1366
shopping        385
verification        697
statement        459
was        64
statements        2233
when        2574
stop        131
withdrawals        6488
tactics        6125
workout        256
taking        3019
wrong        17
terms        115
you        2510
the        3264
your        3409
theft        1347
threatening        2276
to        4402
transaction        738
transfer        197
```

```
unable        4179
underwriting        951
unsolicited        280
use        819
using        1056
verification        4517
was        210
when        1521
with        1944
withdrawals        4067
workout        94
wrong        152
you        1311
your        435
```

The mapper and reducer counts were 2 and 1 respectively, which are the defaults for Hadoop Streaming. The combiner counter was 2. This is somewhat strange as there were certainly more than 2 different unique keys emitted by the mappers. However, Hadoop does not guarantee that the combiner will run a certain number of times.

1.4.4 HW 3.2.0.4

Using a single reducer: What are the top 50 most frequent terms in your word count analysis? Present the top 50 terms and their frequency and their relative frequency. Present the top 50 terms and their frequency and their relative frequency. If there are ties please sort the tokens in alphanumeric/string order. Present bottom 10 tokens (least frequent items).

In [29]:
```
%%writefile mapper.py
#!/usr/bin/python
# mapper.py
# Author:Jackson Lane
# Description: mapper code for HW3.2.0.4
from __future__ import print_function

import sys,re
sys.stderr.write("reporter:counter:3.2,MapperCount,1\n")
for line in sys.stdin:
    fields = line.split(",")
    issue = fields[3]
    words = re.findall("[\w']+",issue)
    for word in words:
        word = word.lower()
        # I'm emitting the number first here so that I can use order inversion to emit the tota
        print (word,1,sep=",")
```

```
Overwriting mapper.py
```

In [30]:
```
%%writefile reducer.py
#!/usr/bin/python
# reducer.py
# Author:Jackson Lane
# Description: reducer code for HW3.2.0.4
# Since we are using a single reducer, it has to sum up the word counts and get the top 50 and
```

```
        import sys,Queue
        from collections import deque
        words = {}
        sys.stderr.write("reporter:counter:3.2,ReducerCount,1\n")
        total = float(0)
        min_elements= []
        max_elements = deque([])
        for line in sys.stdin:
                #Parse in the word and count from the mapper
                word,value = line.split(",")
                word = word
                value = int(value)
                #Update the master word count dictionary
                words[word] = words.setdefault(word, 0) + value
                #Update the total number of words as well
                total += value
        #Convert the words dictionary into an array of tuples
        words =[(k, v) for k, v in words.iteritems()]
        #Sort the array of tuples by count and then alphabetically by word.
        words   = sorted(words,key=lambda x: x[0])
        words   = sorted(words,key=lambda x: x[1])

        #Get the top 50 and bottom elements from the list of tuples
        for (word,value) in words:
                value = float(value)
                if len(min_elements) < 10:
                    min_elements.append((word,value,value/total))
                if(len(max_elements) >= 50):
                    max_elements.popleft()
                max_elements.append((word,value,value/total))

        #Print out the max 50 and min 10 elements
        print "50 most common tokens:"
        for i,p in enumerate(max_elements):
                print str(50-i)+":",p
        print "10 least common tokens:"
        for i,p in enumerate(min_elements):
                print str(i+1)+":",p
```

```
Overwriting reducer.py
```

```
In [31]: !hdfs dfs -rm -r results/3.2
         !hdfs dfs -put Consumer_Complaints.csv
         !hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
         -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCompara
         -D mapreduce.partition.keycomparator.options=-n \
         -D mapreduce.output.key.field.separator="," \
         -D mapred.reduce.tasks=1 \
         -file mapper.py \
         -file reducer.py \
         -mapper "mapper.py" \
         -reducer "reducer.py" \
         -input Consumer_Complaints.csv \
         -output results/3.2
```

```
16/06/05 11:11:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 11:11:46 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minut
Deleted results/3.2
16/06/05 11:11:48 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
put: ‘Consumer_Complaints.csv’: File exists
16/06/05 11:11:49 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/06/05 11:11:49 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-unjar3310236814296116326/] [] /tmp/streamjob661080970
16/06/05 11:11:50 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:11:50 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:11:51 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 11:11:51 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 11:11:51 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapred
16/06/05 11:11:51 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0189
16/06/05 11:11:52 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0189
16/06/05 11:11:52 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicati
16/06/05 11:11:52 INFO mapreduce.Job: Running job: job_1464324416493_0189
16/06/05 11:11:59 INFO mapreduce.Job: Job job_1464324416493_0189 running in uber mode : false
16/06/05 11:11:59 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 11:12:07 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 11:12:17 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 11:12:17 INFO mapreduce.Job: Job job_1464324416493_0189 completed successfully
16/06/05 11:12:17 INFO mapreduce.Job: Counters: 52
        File System Counters
                FILE: Number of bytes read=12213953
                FILE: Number of bytes written=24791994
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50910610
                HDFS: Number of bytes written=2833
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=48336
                Total time spent by all reduces in occupied slots (ms)=57896
                Total time spent by all map tasks (ms)=12084
                Total time spent by all reduce tasks (ms)=7237
                Total vcore-milliseconds taken by all map tasks=12084
                Total vcore-milliseconds taken by all reduce tasks=7237
                Total megabyte-milliseconds taken by all map tasks=49496064
                Total megabyte-milliseconds taken by all reduce tasks=59285504
        Map-Reduce Framework
                Map input records=312912
                Map output records=980482
                Map output bytes=10252983
                Map output materialized bytes=12213959
                Input split bytes=202
                Combine input records=0
```

```
                Combine output records=0
                Reduce input groups=593442
                Reduce shuffle bytes=12213959
                Reduce input records=980482
                Reduce output records=62
                Spilled Records=1960964
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=196
                CPU time spent (ms)=9020
                Physical memory (bytes) snapshot=780902400
                Virtual memory (bytes) snapshot=19000471552
                Total committed heap usage (bytes)=644874240
        3.2
                MapperCount=2
                ReducerCount=1
        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=50910408
        File Output Format Counters
                Bytes Written=2833
16/06/05 11:12:17 INFO streaming.StreamJob: Output directory: results/3.2
```

```
In [32]: !hdfs dfs -cat results/3.2/part-00000
```

```
16/06/05 11:12:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
50 most common tokens:
50: ('score', 4357.0, 0.004443732776328377)
49: ('card', 4405.0, 0.004492688290045101)
48: ('identity', 4729.0, 0.0048231380076329804)
47: ("company's", 4858.0, 0.0049547059507466734)
46: ('investigation', 4858.0, 0.0049547059507466734)
45: ('managing', 5006.0, 0.005105652118039903)
44: ('disclosure', 5214.0, 0.0053177926774790355)
43: ('verification', 5214.0, 0.0053177926774790355)
42: ('process', 5505.0, 0.005614585479386669)
41: ('being', 5663.0, 0.0057757307120375485)
40: ('by', 5663.0, 0.0057757307120375485)
39: ('caused', 5663.0, 0.0057757307120375485)
38: ('funds', 5663.0, 0.0057757307120375485)
37: ('low', 5663.0, 0.0057757307120375485)
36: ('the', 6248.0, 0.0063723760354601105)
35: ('lease', 6337.0, 0.006463147717143201)
34: ('reporting', 6559.0, 0.006689566968083045)
33: ('communication', 6920.0, 0.007057753227494233)
32: ('tactics', 6920.0, 0.007057753227494233)
31: ('disputes', 6938.0, 0.007076111545138004)
30: ('other', 7886.0, 0.008042982941043282)
```

```
29: ('billing', 8158.0, 0.00832039751877138)
28: ('unable', 8178.0, 0.008340795649486681)
27: ('to', 8401.0, 0.00856823480696229)
26: ('application', 8868.0, 0.009044531159164574)
25: ('problems', 9484.0, 0.009672793585195853)
24: ('deposits', 10555.0, 0.010765113485000234)
23: ('withdrawals', 10555.0, 0.010765113485000234)
22: ('my', 10731.0, 0.010944617035294885)
21: ('of', 10885.0, 0.011101682641802705)
20: ('attempts', 11848.0, 0.01208385263574446)
19: ('collect', 11848.0, 0.01208385263574446)
18: ("cont'd", 11848.0, 0.01208385263574446)
17: ('owed', 11848.0, 0.01208385263574446)
16: ('not', 12353.0, 0.012598905436305817)
15: ('club', 12545.0, 0.012794727491172709)
14: ('health', 12545.0, 0.012794727491172709)
13: ('opening', 16205.0, 0.01652758541207284)
12: ('and', 16448.0, 0.016775422700263748)
11: ('debt', 19309.0, 0.01969337529908759)
10: ('account', 20681.0, 0.02109268706615726)
9: ('or', 22533.0, 0.022981553970394152)
8: ('information', 29069.0, 0.029647663088154603)
7: ('on', 29069.0, 0.029647663088154603)
6: ('incorrect', 29133.0, 0.029712937106443564)
5: ('report', 34903.0, 0.035597797817807975)
4: ('servicing', 36767.0, 0.03749890360047405)
3: ('credit', 55251.0, 0.05635085600755547)
2: ('modification', 70487.0, 0.07189015198647196)
1: ('loan', 119630.0, 0.12201141887357443)
10 least common tokens:
1: ('disclosures', 64.0, 6.5274018288964e-05)
2: ('missing', 64.0, 6.5274018288964e-05)
3: ('amt', 71.0, 7.241336403931944e-05)
4: ('day', 71.0, 7.241336403931944e-05)
5: ('checks', 75.0, 7.649299018237969e-05)
6: ('convenience', 75.0, 7.649299018237969e-05)
7: ('credited', 92.0, 9.383140129038574e-05)
8: ('payment', 92.0, 9.383140129038574e-05)
9: ('amount', 98.0, 9.995084050497613e-05)
10: ('apply', 118.0, 0.00012034897122027737)
```

1.4.5 HW 3.2.1

Using 2 reducers: What are the top 50 most frequent terms in your word count analysis? Present the top 50 terms and their frequency and their relative frequency. Present the top 50 terms and their frequency and their relative frequency. If there are ties please sort the tokens in alphanumeric/string order. Present bottom 10 tokens (least frequent items). Please use a combiner.

For this problem, my understanding from the discussion board was to use a single mapreduce job with a single reducer.py with two reducer tasks. I understand that other students interpreted the problem as meaning they could use two different reducer.py files across two different map reduce jobs. I also understand that other students interpreted the problem as meaning they could feed in the wordcounts from the previous parts of 3.2 into this problem. Both of these methods might have been easier, but I wanted to see if I could perform the word count and get the top 50 and bottom 10 elements with just a single map reduce job starting from the Consumer_Complaints.csv.

The challenging part was getting the top 50 and bottom 10 elements into different partitions while still keeping in the spirit of MapReduce. There's not enough information at the mapper stage to determine whether a word is in the top 50 or bottom 10. One could put some reducer functionality in the mapper to first aggregate the counts of each word and then determine the partition key, but that goes against the intended purpose of a mapper.

To work around this, I had my mappers emit each word twice (once for each partition). Normally, this would defeat the purpose of having partitions as each reducer would need to process the full dataset. However, I also wrote my combiners to filter the mapper outputs after aggregation so that only the high count words make it to the top 50 reducer task and only the low count words make it to the bottom 10 reducer task. This type of solution also fits with the combiners expressed purpose of reducing the volume of data sent to the reducers.

While my solution gets the top 50 and bottom 10 words, the accurarcy drops beyond those ranks (for example, it's not accurate on the top 100 and bottom 50 words). Since a combiner may be applied on just a subset of the keys emitted by the mapper, the combiner cannot know for sure whether a word appears with a high or low frequency relative to the other words in the corpus. In fact, the combiner cannot even know for sure the total number of words in the corpus as it might not be processing all the words. The reducer does not have this problem because it has barrier synchronization.

In [33]:
```
%%writefile mapper.py
#!/usr/bin/python
# mapper.py
# Author:Jackson Lane
# Description: mapper code for HW3.2.1
from __future__ import print_function
from sets import Set
import sys,re,random
sys.stderr.write("reporter:counter:3.2,MapperCount,1\n")
total = 0

uniquewords = Set([])
for line in sys.stdin:
    fields = line.split(",")
    issue = fields[3]
    words = re.findall("[\w']+",issue)
    for word in words:
        word = word.lower()
        uniquewords.add(word)
        total+= 1
        # I'm emitting the number first here so that I can use order inversion to emit the tota
        print(0,1,word,sep="\t")
        print(1,1,word,sep="\t")


#Since we need to compute relative frequencies in the reducer, we need the total number of issu
#But since reading counters from within a job is apparently bad practice, I'm supposed to print
# a special key value pair with the fields that I need.
# emit total twice for each partition
print(0,0,total,sep="\t")
print(0,-1,uniquewords,sep="\t")
print(1,0,total,sep="\t")
print(1,-1,uniquewords,sep="\t")
```

```
Overwriting mapper.py
```

In [43]:
```
%%writefile combiner.py
#!/usr/bin/python
```

```
# reducer.py
# Author: Jackson Lane
# Description: combiner code for 3.2.1
# Acts like router, only sending the small word counts to the first reducer and the large word
# This type of "filter combiner" currently only works with 2 mappers
# Alternatively I could just remove the filter condition and make it so that this combiner pass
# word counts.  That would be scalable to any number of mappers, but it would defeat the purpos
# as each reducer would process the full word count work load.
# If there are more mappers, then this combiner will end up sending the wrong lines to the wro

from __future__ import print_function
from sets import Set

from operator import itemgetter
import sys
#Output to combiner is sorted and paritioned already, so we can use reducer word count log
partition = 0
word = ""
count = 0
#Since each instance of a combiner only processes a subset of the data emitted by the mappers,
# the distribution of the words observed in the combiner is not always representative of
# the distribution of words in the rest of the corpus.
# But since the we want the top 50 words but only the bottom 10 words, we can err on the side
# of the top 50 since it needs a higher level of accuracy.  The below magic number means that
# we send every word with a count that is in the top 80% to the top 50 reducer and the bottom
# 20% to the bottom 10 reducer
magicnumber = .4
sys.stderr.write("reporter:counter:3.2,CombinerCount,1\n")
total = float(0)
sys.stderr.write("combiner\n")
uniquewords = Set([])

for line in sys.stdin:
    sys.stderr.write(line)
    # remove leading and trailing whitespace
    line = line.strip()
    newpartition, newcount, newword = line.split("\t")
    newcount = int(newcount)
    newpartition = int(newpartition)
    # Update the summary statistic variable we passed along from the mapper
    # These values should appear first in the stdin because they have counts
    # of 0 and -1
    if newcount == 0:
        total += float(newword)
        continue
    if newcount == -1:
        uniquewords.update(eval(newword))
        continue

    #Update the word counts
    if (newword == word and partition == newpartition):
        count += newcount
    else:
        # We have finished with all instances of the current word.
```

```
            # Now determine which partition the word should go to
            partitionkey =int(count> total*magicnumber / len(uniquewords))

            # If word is routed to correct partition, emit word and count.  Otherwise, just skip w
            if (count > 0 and partitionkey == partition): print(partition,count,word, sep='\t')
            word = newword
            count = newcount
            partition = newpartition

    #Emit last word count if going to right partition
    partitionkey =int(count > total*magicnumber  / len(uniquewords))
    if (count > 0 and partitionkey == partition):
        print(partition,count,word, sep='\t')

    #pass on totals again
    print(partitionkey,0,3total,sep="\t")
    #We don't need to pass on the uniquewordcount
```

```
Overwriting combiner.py
```

In [39]:
```
%%writefile reducer.py
#!/usr/bin/python
# reducer.py
# Author:Jackson Lane
# Description: reducer code for HW3.2.1
# This reducer collects all the incoming words into a dictionary
# with their counts and relative frequencies.  Then the reducer
# sorts the dictionary by count and outputs either the
# top 50 or bottom 10 word counts

import sys
words = {}
sys.stderr.write("reporter:counter:3.2,ReducerCount,1\n")
total = float(0)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    p, count, word = line.split("\t")
    p = int(p)
    count = float(count)
    # Update the total summary statistic variable
    # Since combiner output is not sorted this one may
    # no longer be at the top of the stdin.  So
    # that's why we can only compute relative frequency
    # after all the stdin has been processed.
    if count == 0:
        total += float(word)
        continue

    #Update the word count dictionary.
    words[word] = words.setdefault(word, 0) + count

#Get which partition this reducer represents.
# If 0, then it's the bottom 10 reducer.
# If 1, then it's the top 50 reducer.
```

```
        partition = p

        #Turn dictionary into sorted array of tuples with counts and
        # relative frequencies
        words =[(k, v,float(v)/total) for k, v in words.iteritems()]
        words   = sorted(words,key=lambda x: x[0])
        words   = sorted(words,key=lambda x: x[1])

        #Print either top 50 or bottom 10
        if partition == 1:
            print "50 most common tokens:"
            for i,p in enumerate(words[-50:]):
                print str(50-i)+":",p

        else:
            print "10 least common tokens:"
            for i,p in enumerate(words[:10]):
                print str(i+1)+":",p


Overwriting reducer.py

In [40]: !hdfs dfs -rm -r results/3.2
         !hdfs dfs -put Consumer_Complaints.csv
         !hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
         -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedComparat
         -D mapreduce.partition.keycomparator.options="-k2,2n -k3,3" \
         -D mapreduce.partition.keypartitioner.options="-k1,1n" \
         -D stream.num.map.output.key.fields=3 \
         -D mapreduce.job.maps=2 \
         -D mapreduce.job.reduces=2 \
         -file mapper.py \
         -file combiner.py \
         -file reducer.py \
         -mapper "mapper.py" \
         -combiner "combiner.py" \
         -reducer "reducer.py" \
         -partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner \
         -input Consumer_Complaints.csv \
         -output results/3.2

16/06/05 11:15:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 11:15:57 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minut
Deleted results/3.2
16/06/05 11:15:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
put: ‘Consumer_Complaints.csv’: File exists
16/06/05 11:16:00 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/06/05 11:16:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, combiner.py, reducer.py, /tmp/hadoop-unjar1441934325260427106/] [] /tmp/strea
16/06/05 11:16:01 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:16:01 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:16:02 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 11:16:02 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 11:16:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0191
16/06/05 11:16:03 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0191
```

```
16/06/05 11:16:03 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicati
16/06/05 11:16:03 INFO mapreduce.Job: Running job: job_1464324416493_0191
16/06/05 11:16:10 INFO mapreduce.Job: Job job_1464324416493_0191 running in uber mode : false
16/06/05 11:16:10 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 11:16:21 INFO mapreduce.Job:  map 67% reduce 0%
16/06/05 11:16:29 INFO mapreduce.Job:  map 83% reduce 0%
16/06/05 11:16:30 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 11:16:36 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 11:16:36 INFO mapreduce.Job: Job job_1464324416493_0191 completed successfully
16/06/05 11:16:36 INFO mapreduce.Job: Counters: 53
        File System Counters
                FILE: Number of bytes read=5130
                FILE: Number of bytes written=499622
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50910610
                HDFS: Number of bytes written=2831
                HDFS: Number of read operations=12
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=2
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=140068
                Total time spent by all reduces in occupied slots (ms)=47752
                Total time spent by all map tasks (ms)=35017
                Total time spent by all reduce tasks (ms)=5969
                Total vcore-milliseconds taken by all map tasks=35017
                Total vcore-milliseconds taken by all reduce tasks=5969
                Total megabyte-milliseconds taken by all map tasks=143429632
                Total megabyte-milliseconds taken by all reduce tasks=48898048
        Map-Reduce Framework
                Map input records=312912
                Map output records=1960972
                Map output bytes=24434674
                Map output materialized bytes=5142
                Input split bytes=202
                Combine input records=1960972
                Combine output records=317
                Reduce input groups=5
                Reduce shuffle bytes=5142
                Reduce input records=317
                Reduce output records=62
                Spilled Records=634
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=369
                CPU time spent (ms)=33610
                Physical memory (bytes) snapshot=1195876352
                Virtual memory (bytes) snapshot=27568566272
```

```
                Total committed heap usage (bytes)=1041760256
        3.2
                CombinerCount=4
                MapperCount=2
                ReducerCount=2
        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=50910408
        File Output Format Counters
                Bytes Written=2831
16/06/05 11:16:36 INFO streaming.StreamJob: Output directory: results/3.2
```

```
In [42]: !hdfs dfs -cat results/3.2/part-0000*
```

```
16/06/05 11:17:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
10 least common tokens:
1: ('disclosures', 64.0, 6.5274018288964e-05)
2: ('missing', 64.0, 6.5274018288964e-05)
3: ('amt', 71.0, 7.241336403931944e-05)
4: ('day', 71.0, 7.241336403931944e-05)
5: ('checks', 75.0, 7.649299018237969e-05)
6: ('convenience', 75.0, 7.649299018237969e-05)
7: ('credited', 92.0, 9.383140129038574e-05)
8: ('payment', 92.0, 9.383140129038574e-05)
9: ('amount', 98.0, 9.995084050497613e-05)
10: ('apply', 118.0, 0.00012034897122027737)
50 most common tokens:
50: ('score', 4357.0, 0.004443732776328377)
49: ('card', 4405.0, 0.004492688290045101)
48: ('disclosure', 4517.0, 0.004606917822050787)
47: ('verification', 4517.0, 0.004606917822050787)
46: ('identity', 4729.0, 0.0048231380076329804)
45: ("company's", 4858.0, 0.0049547059507466734)
44: ('investigation', 4858.0, 0.0049547059507466734)
43: ('managing', 5006.0, 0.005105652118039903)
42: ('process', 5505.0, 0.005614585479386669)
41: ('being', 5663.0, 0.0057757307120375485)
40: ('by', 5663.0, 0.0057757307120375485)
39: ('caused', 5663.0, 0.0057757307120375485)
38: ('funds', 5663.0, 0.0057757307120375485)
37: ('low', 5663.0, 0.0057757307120375485)
36: ('communication', 6125.0, 0.006246927531561008)
35: ('tactics', 6125.0, 0.006246927531561008)
34: ('the', 6248.0, 0.0063723760354601105)
33: ('lease', 6337.0, 0.006463147717143201)
32: ('reporting', 6559.0, 0.006689566968083045)
31: ('disputes', 6938.0, 0.007076111545138004)
30: ('other', 7886.0, 0.008042982941043282)
29: ('billing', 8158.0, 0.00832039751877138)
```

```
28: ('unable', 8178.0, 0.008340795649486681)
27: ('to', 8401.0, 0.00856823480696229)
26: ('application', 8868.0, 0.009044531159164574)
25: ('problems', 9484.0, 0.009672793585195853)
24: ('deposits', 10555.0, 0.010765113485000234)
23: ('withdrawals', 10555.0, 0.010765113485000234)
22: ('my', 10731.0, 0.010944617035294885)
21: ('of', 10885.0, 0.011101682641802705)
20: ('attempts', 11848.0, 0.01208385263574446)
19: ('collect', 11848.0, 0.01208385263574446)
18: ("cont'd", 11848.0, 0.01208385263574446)
17: ('owed', 11848.0, 0.01208385263574446)
16: ('not', 12353.0, 0.012598905436305817)
15: ('club', 12545.0, 0.012794727491172709)
14: ('health', 12545.0, 0.012794727491172709)
13: ('opening', 16205.0, 0.01652758541207284)
12: ('and', 16448.0, 0.016775422700263748)
11: ('debt', 19309.0, 0.01969337529908759)
10: ('account', 20681.0, 0.02109268706615726)
9: ('or', 22533.0, 0.022981553970394152)
8: ('information', 29069.0, 0.029647663088154603)
7: ('on', 29069.0, 0.029647663088154603)
6: ('incorrect', 29133.0, 0.029712937106443564)
5: ('report', 34903.0, 0.035597797817807975)
4: ('servicing', 36767.0, 0.03749890360047405)
3: ('credit', 55251.0, 0.05635085600755547)
2: ('modification', 70487.0, 0.07189015198647196)
1: ('loan', 119630.0, 0.12201141887357443)
```

1.5 HW3.3. Shopping Cart Analysis

For this homework use the online browsing behavior dataset located at:

`https://www.dropbox.com/s/zlfyiwa70poqg74/ProductPurchaseData.txt?dl=0`

Do some exploratory data analysis of this dataset.

How many unique items are available from this supplier?

Using a single reducer: Report your findings such as number of unique products; largest basket; report the top 50 most frequently purchased items, their frequency, and their relative frequency (break ties by sorting the products alphabetical order) etc. using Hadoop Map-Reduce.

```
In [44]: %%writefile mapper.py
         #!/usr/bin/python
         # mapper.py
         # Author:Jackson Lane
         # Description: mapper code for HW3.3
         from __future__ import print_function
         from sets import Set
         import sys,re
         sys.stderr.write("reporter:counter:3.3,MapperCount,1\n")
         total = 0
         #Maintain a set of unique products to pass to reducer
         uniqueproducts = Set([])
         for line in sys.stdin:
             basket = re.findall("[\w']+",line)
```

```
            for product in basket:
                uniqueproducts.add(product)
                total+= 1
                #Emit three fields for the reducer to use:
                # Product, count, and basket size
                # Basket size will be emitted once for each
                # product in the basket.  This is inefficient
                # from a network bandwidth perspective, but
                # relatively harmless as it's just an
                # extra integer.
                # If we really wanted to save bandwidth, we
                # could just emit the product by itself and
                # nothing else.  The reducer could assume that
                # the count will be 1.
                print (product,1,len(basket),sep="\t")


        #Emit the total number of products and the set of unique products to reducer
        print('Totals',total,uniqueproducts,sep="\t")
```

```
Overwriting mapper.py
```

```
In [45]: %%writefile reducer.py
         #!/usr/bin/python
         # reducer.py
         # Author:Jackson Lane
         # Description: reducer code for HW3.3
         # This reducer gets the products from the mapper and computes
         # The number of unique products, the largest basket, and the
         # top 50 products.
         # Since we are using a single reducer, we have more flexibility
         # with what we are allowed to do.

         import sys
         from sets import Set

         sys.stderr.write("reporter:counter:3.3,ReducerCount,1\n")
         total = float(0)
         uniqueproducts = Set([])

         # These fields are similar to the fields from the word count
         # code, but renamed for products instead of words
         product = 0
         maxbasketsize = 0
         count =0
         products = {}

         for line in sys.stdin:
             line = line.strip()

             #Get the product, count, and basket size
             newproduct,newcount,basketsize=line.split("\t")
             newcount = int(newcount)

             # Get the total products and the set of unique products.
```

```
            # These fields do not have to come first as we compute
            # the relative frequencies after all stdin has been
            # processed
            if newproduct=="Totals":
                total+=newcount
                uniqueproducts.update(eval(basketsize))
                continue
            basketsize=int(basketsize)

            #Update maximum basket size
            if basketsize>maxbasketsize:
                maxbasketsize=basketsize

            if product==newproduct:
                count+=newcount
            else:
                # We are finished with the current product
                # Update the products dictionary with the product count
                products[product] = products.setdefault(product, 0) + count
                product=newproduct
                count=newcount

        # Change products dictionary in an array of tuples with counts
        # and relative frequencies
        products =[(k, v,float(v)/total) for k, v in products.iteritems()]

        # Sort the dictionary by count and then by product name
        products   = sorted(products,key=lambda x: x[0])
        products   = sorted(products,key=lambda x: x[1])

        # Output the number of unique products, largest basket size, and top 50
        # products.
        print "Number of Unique Products:",len(uniqueproducts)
        print "Largest Basket Size:", maxbasketsize
        print "Top 50 Products:"
        for i,p in enumerate(products[-50:]):
            print str(50-i)+":",p
```

```
Overwriting reducer.py
```

Run the mapreduce job

```
In [46]: !hdfs dfs -rm -r results/3.3

        !hdfs dfs -put -p -f ProductPurchaseData.txt
        !hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
        -file mapper.py \
        -file reducer.py \
        -mapper "mapper.py" \
        -reducer "reducer.py" \
        -input ProductPurchaseData.txt \
        -output results/3.3
```

```
16/06/05 11:56:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
rm: ‘results/3.3’: No such file or directory
```

```
16/06/05 11:56:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 11:56:29 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/06/05 11:56:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-unjar1357311333018480171/] [] /tmp/streamjob72353093
16/06/05 11:56:29 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:56:30 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 11:56:31 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 11:56:31 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 11:56:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0192
16/06/05 11:56:31 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0192
16/06/05 11:56:31 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicat
16/06/05 11:56:31 INFO mapreduce.Job: Running job: job_1464324416493_0192
16/06/05 11:56:39 INFO mapreduce.Job: Job job_1464324416493_0192 running in uber mode : false
16/06/05 11:56:39 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 11:56:48 INFO mapreduce.Job:  map 50% reduce 0%
16/06/05 11:56:49 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 11:56:56 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 11:56:57 INFO mapreduce.Job: Job job_1464324416493_0192 completed successfully
16/06/05 11:56:57 INFO mapreduce.Job: Counters: 52
        File System Counters
                FILE: Number of bytes read=6230138
                FILE: Number of bytes written=12822861
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3462815
                HDFS: Number of bytes written=2380
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=60796
                Total time spent by all reduces in occupied slots (ms)=45096
                Total time spent by all map tasks (ms)=15199
                Total time spent by all reduce tasks (ms)=5637
                Total vcore-milliseconds taken by all map tasks=15199
                Total vcore-milliseconds taken by all reduce tasks=5637
                Total megabyte-milliseconds taken by all map tasks=62255104
                Total megabyte-milliseconds taken by all reduce tasks=46178304
        Map-Reduce Framework
                Map input records=31101
                Map output records=380826
                Map output bytes=5468474
                Map output materialized bytes=6230144
                Input split bytes=202
                Combine input records=0
                Combine output records=0
                Reduce input groups=12593
                Reduce shuffle bytes=6230144
                Reduce input records=380826
```

```
                Reduce output records=53
                Spilled Records=761652
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=316
                CPU time spent (ms)=5940
                Physical memory (bytes) snapshot=699662336
                Virtual memory (bytes) snapshot=18995019776
                Total committed heap usage (bytes)=619184128
        3.3
                MapperCount=2
                ReducerCount=1
        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=3462613
        File Output Format Counters
                Bytes Written=2380
16/06/05 11:56:57 INFO streaming.StreamJob: Output directory: results/3.3
```

```
In [47]: !hdfs dfs -cat results/3.3/part-00000
```

```
16/06/05 11:56:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Number of Unique Products: 12592
Largest Basket Size: 37
Top 50 Products:
50: ('GRO85051', 1214, 0.0031878242967880175)
49: ('DAI22896', 1219, 0.0032009537214041134)
48: ('GRO81087', 1220, 0.0032035796063273323)
47: ('DAI31081', 1261, 0.0033112408881793166)
46: ('GRO15017', 1275, 0.003348003277104384)
45: ('ELE91337', 1289, 0.0033847656660294517)
44: ('DAI43223', 1290, 0.003387391550952671)
43: ('SNA96271', 1295, 0.0034005209755687666)
42: ('ELE59935', 1311, 0.003442535134340273)
41: ('DAI88807', 1316, 0.0034556645589563684)
40: ('ELE74482', 1316, 0.0034556645589563684)
39: ('GRO61133', 1321, 0.003468793983572464)
38: ('ELE56788', 1345, 0.003531815221729723)
37: ('GRO38814', 1352, 0.0035501964161922567)
36: ('SNA90094', 1390, 0.0036499800432745837)
35: ('SNA93860', 1407, 0.0036946200869693085)
34: ('FRO53271', 1420, 0.003728756590971157)
33: ('FRO35904', 1436, 0.0037707707497426635)
32: ('ELE34057', 1489, 0.003909942650673277)
31: ('GRO94758', 1489, 0.003909942650673277)
30: ('ELE99737', 1516, 0.003980841543600193)
29: ('FRO78087', 1531, 0.00402022981744848)
28: ('DAI22177', 1627, 0.004272314770077516)
```

```
27: ('SNA55762', 1646, 0.00432220658361868)
26: ('ELE66810', 1697, 0.0044561267147028545)
25: ('FRO32293', 1702, 0.004469256139318951)
24: ('DAI83733', 1712, 0.004495514988551142)
23: ('ELE66600', 1713, 0.004498140873474361)
22: ('GRO46854', 1756, 0.004611053925172783)
21: ('DAI63921', 1773, 0.004655693968867509)
20: ('GRO56726', 1784, 0.0046845787030229185)
19: ('ELE74009', 1816, 0.004768607020565931)
18: ('GRO30386', 1840, 0.00483162825872319)
17: ('FRO85978', 1918, 0.005036447282734282)
16: ('GRO71621', 1920, 0.005041699052580719 5)
15: ('GRO59710', 2004, 0.005262273386131126)
14: ('SNA99873', 2083, 0.005469718295065437)
13: ('GRO21487', 2115, 0.005553746612608449)
12: ('FRO80039', 2233, 0.005863601033548306)
11: ('ELE26917', 2292, 0.006018528244018234)
10: ('DAI85309', 2293, 0.006021154128941453)
9: ('FRO31317', 2330, 0.006118311871100561)
8: ('SNA45677', 2455, 0.006446547486502951)
7: ('DAI75645', 2736, 0.007184421149927526)
6: ('ELE32164', 2851, 0.007486397916097725)
5: ('SNA80324', 3044, 0.007993193706279015)
4: ('GRO73461', 3602, 0.009458437493435288)
3: ('ELE17451', 3875, 0.010175304077474108)
2: ('FRO40251', 3881, 0.010191059387013424)
1: ('DAI62779', 6667, 0.017506774783101905)
```

1.6 HW3.4. Pairs

Suppose we want to recommend new products to the customer based on the products they have already browsed on the online website. Write a map-reduce program to find products which are frequently browsed together. Fix the support count (cooccurence count) to s = 100 (i.e. product pairs need to occur together at least 100 times to be considered frequent) and find pairs of items (sometimes referred to itemsets of size 2 in association rule mining) that have a support count of 100 or more.

List the top 50 product pairs with corresponding support count (aka frequency), and relative frequency or support (number of records where they coccur, the number of records where they coccur/the number of baskets in the dataset) in decreasing order of support for frequent (100>count) itemsets of size 2.

Use the Pairs pattern (lecture 3) to extract these frequent itemsets of size 2. Free free to use combiners if they bring value. Instrument your code with counters for count the number of times your mapper, combiner and reducers are called.

Please output records of the following form for the top 50 pairs (itemsets of size 2):

```
item1, item2, support count, support
```

Fix the ordering of the pairs lexicographically (left to right), and break ties in support (between pairs, if any exist) by taking the first ones in lexicographically increasing order.

Report the compute time for the Pairs job. Describe the computational setup used (E.g., single computer; dual core; linux, number of mappers, number of reducers) Instrument your mapper, combiner, and reducer to count how many times each is called using Counters and report these counts.

```
In [123]: %%writefile mapper.py
          #!/usr/bin/python

          #HW 3.4 - Mapper Function Code
```

```
from __future__ import print_function
from sets import Set

import sys,re
sys.stderr.write("reporter:counter:3.4,MapperCount,1\n")

baskets =0

for line in sys.stdin:
    line=line.strip()
    # Using a set to avoid double counting in instances where a customer
    # bought more than 1 of a product
    basket = list(Set(re.findall("[\w']+",line)))
    baskets +=1
    for i,p1 in enumerate(basket):
        # Only iterate through products we haven't seen yet to avoid
        # duplicates
        for p2 in basket[i+1:]:
            # Create pairs out of the two products.  For consitency,
            # put the alphbetically higher element first in the pair.
            # Then emit both products and the count
            if p2 > p1:
                print (p1,p2,1,sep='\t')
            else:
                print (p2,p1,1,sep='\t')

# Output total number of baskets with a leading space in the key for
# order-inversion purposes
print (" Total","Baskets",baskets,sep='\t')
```

```
Overwriting mapper.py
```

In [124]:
```
%%writefile combiner.py
#!/usr/bin/python
# combiner.py
# Author:Jackson Lane
# Description: combiner code for HW3.4
from __future__ import print_function

import sys

sys.stderr.write("reporter:counter:3.4,Combiner,1\n")
product1 =''
product2 = ''
count =0

for line in sys.stdin:
    line = line.strip()

    # Parse the two product fields and the count field from the mapper
    newproduct1,newproduct2,newcount=line.split("\t")
    newcount = int(newcount)

    # Pass along the total number of baskets to the reducer
    if newproduct1=="Total":
```

```
                print (" Total","Baskets",newcount,sep='\t')
                continue
            if product1 == newproduct1 and product2 == newproduct2:
                count+=newcount
            else:
                if(count > 0): print (product1,product2,count,sep='\t')
                product1=newproduct1
                product2=newproduct2
                count=newcount

        if(count > 0): print (product1,product2,count,sep='\t')
```

```
Overwriting combiner.py
```

In [125]:
```
%%writefile reducer.py
#!/usr/bin/python
# reducer.py
# Author:Jackson Lane
# Description: reducer code for HW3.2.4
# Since we are using a single reducer, it has to sum up the word counts and get the top 50 an
from __future__ import print_function

import sys

sys.stderr.write("reporter:counter:3.4,ReducerCount,1\n")
total = float(0)
baskets = float(0)
pair =('','')
count =0
pairs = {}

for line in sys.stdin:
    line = line.strip()
    # Parse the two product fields and the count field from the mapper
    newproduct1,newproduct2,newcount=line.split("\t")
    newcount = int(newcount)
    #Extract the total number of baskets
    #This field should come first in the stdin due to order inversion
    if newproduct1=="Total":
        baskets+=newcount
        continue
    # Turn the two products into a tuple.  This is more for convience
    # than functionality, as it's easier to compare two tuples than
    # four values.  Note that we will end up outputting the two products
    # separately
    newpair = (newproduct1,newproduct2)
    if (newpair == pair):
        count+=newcount
    else:
        #Only consider products that were purchased more than 100 times
        if (count > 100):
            print (pair[0],pair[1],count,count / baskets, sep = ",")
        pair=newpair
        count=newcount
```

```
        #Emit the last product and count (if its over 100 of courses)
        if (count > 100):  print (pair[0],pair[1],count,count / baskets, sep = ",")
```

```
Overwriting reducer.py
```

Run the mapreduce job

```
In [126]: !hdfs dfs -rm -r temp

        !hdfs dfs -put -p -f ProductPurchaseData.txt
        !hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
        -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCompara
        -D mapreduce.partition.keycomparator.options='-k1,2' \
        -D stream.num.map.output.key.fields=2 \
        -D stream.num.reduce.output.key.fields=2 \
        -file mapper.py \
        -file combiner.py \
        -file reducer.py \
        -mapper "mapper.py" \
        -combiner "combiner.py" \
        -reducer "reducer.py" \
        -input ProductPurchaseData.txt \
        -output temp

        !hdfs dfs -rm -r results/3.4

        # Second map reduce job to sort the output.
        !hdfs dfs -put -p -f ProductPurchaseData.txt
        !hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
        -D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCompara
        -D mapreduce.partition.keycomparator.options='-k3,3nr -k1,2' \
        -D mapreduce.output.key.field.separator="," \
        -D stream.map.output.field.separator=, \
        -D stream.reduce.output.field.separator=, \
        -D stream.map.input.field.separator=, \
        -D stream.reduce.input.field.separator=, \
        -D map.output.key.field.separator=, \
        -D stream.num.map.output.key.fields=4 \
        -D stream.num.reduce.output.key.fields=4 \
        -mapper cat \
        -reducer cat \
        -input temp/part-* \
        -output results/3.4
```

```
16/06/05 13:19:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:19:39 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minut
Deleted temp
16/06/05 13:19:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:19:43 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/06/05 13:19:43 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, combiner.py, reducer.py, /tmp/hadoop-unjar2150924994250287621/] [] /tmp/strea
16/06/05 13:19:44 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:19:44 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:19:45 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 13:19:45 INFO mapreduce.JobSubmitter: number of splits:2
```

```
16/06/05 13:19:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0230
16/06/05 13:19:45 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0230
16/06/05 13:19:45 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicat
16/06/05 13:19:45 INFO mapreduce.Job: Running job: job_1464324416493_0230
16/06/05 13:19:52 INFO mapreduce.Job: Job job_1464324416493_0230 running in uber mode : false
16/06/05 13:19:52 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 13:20:02 INFO mapreduce.Job:  map 33% reduce 0%
16/06/05 13:20:03 INFO mapreduce.Job:  map 67% reduce 0%
16/06/05 13:20:12 INFO mapreduce.Job:  map 83% reduce 0%
16/06/05 13:20:14 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 13:20:22 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 13:20:23 INFO mapreduce.Job: Job job_1464324416493_0230 completed successfully
16/06/05 13:20:23 INFO mapreduce.Job: Counters: 53
        File System Counters
                FILE: Number of bytes read=22624203
                FILE: Number of bytes written=45614783
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3462815
                HDFS: Number of bytes written=52179
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=145148
                Total time spent by all reduces in occupied slots (ms)=69496
                Total time spent by all map tasks (ms)=36287
                Total time spent by all reduce tasks (ms)=8687
                Total vcore-milliseconds taken by all map tasks=36287
                Total vcore-milliseconds taken by all reduce tasks=8687
                Total megabyte-milliseconds taken by all map tasks=148631552
                Total megabyte-milliseconds taken by all reduce tasks=71163904
        Map-Reduce Framework
                Map input records=31101
                Map output records=2534016
                Map output bytes=50680322
                Map output materialized bytes=22624209
                Input split bytes=202
                Combine input records=2534016
                Combine output records=1026709
                Reduce input groups=877096
                Reduce shuffle bytes=22624209
                Reduce input records=1026709
                Reduce output records=1311
                Spilled Records=2053418
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=636
```

```
                CPU time spent (ms)=32120
                Physical memory (bytes) snapshot=1332830208
                Virtual memory (bytes) snapshot=18996748288
                Total committed heap usage (bytes)=1266679808
        3.4
                Combiner=2
                MapperCount=2
                ReducerCount=1
        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=3462613
        File Output Format Counters
                Bytes Written=52179
16/06/05 13:20:23 INFO streaming.StreamJob: Output directory: temp
16/06/05 13:20:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:20:25 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minut
Deleted results/3.4
16/06/05 13:20:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:20:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [/tmp/hadoop-unjar2933380005635235536/] [] /tmp/streamjob2570496342596996042.jar tmpDir=
16/06/05 13:20:29 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:20:30 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:20:30 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 13:20:30 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 13:20:31 INFO Configuration.deprecation: map.output.key.field.separator is deprecated. Instead
16/06/05 13:20:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0231
16/06/05 13:20:31 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0231
16/06/05 13:20:31 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicat
16/06/05 13:20:31 INFO mapreduce.Job: Running job: job_1464324416493_0231
16/06/05 13:20:38 INFO mapreduce.Job: Job job_1464324416493_0231 running in uber mode : false
16/06/05 13:20:38 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 13:20:44 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 13:20:50 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 13:20:50 INFO mapreduce.Job: Job job_1464324416493_0231 completed successfully
16/06/05 13:20:50 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=56118
                FILE: Number of bytes written=475145
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=56461
                HDFS: Number of bytes written=53490
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
```

```
                Launched reduce tasks=1
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=23944
                Total time spent by all reduces in occupied slots (ms)=22920
                Total time spent by all map tasks (ms)=5986
                Total time spent by all reduce tasks (ms)=2865
                Total vcore-milliseconds taken by all map tasks=5986
                Total vcore-milliseconds taken by all reduce tasks=2865
                Total megabyte-milliseconds taken by all map tasks=24518656
                Total megabyte-milliseconds taken by all reduce tasks=23470080
        Map-Reduce Framework
                Map input records=1311
                Map output records=1311
                Map output bytes=53490
                Map output materialized bytes=56124
                Input split bytes=186
                Combine input records=0
                Combine output records=0
                Reduce input groups=1311
                Reduce shuffle bytes=56124
                Reduce input records=1311
                Reduce output records=1311
                Spilled Records=2622
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=206
                CPU time spent (ms)=2160
                Physical memory (bytes) snapshot=691863552
                Virtual memory (bytes) snapshot=18991587328
                Total committed heap usage (bytes)=643301376
        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=56275
        File Output Format Counters
                Bytes Written=53490
16/06/05 13:20:50 INFO streaming.StreamJob: Output directory: results/3.4
```

```
In [122]: !hdfs dfs -cat results/3.4/part-00000 | head -50
```

```
16/06/05 13:19:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
DAI62779,ELE17451,1592,0.0511880646925
FRO40251,SNA80324,1412,0.0454004694383
DAI75645,FRO40251,1254,0.0403202469374
FRO40251,GRO85051,1213,0.0390019613517
DAI62779,GRO73461,1139,0.0366226166361
DAI75645,SNA80324,1130,0.0363332368734
DAI62779,FRO40251,1070,0.0344040384554
```

```
DAI62779,SNA80324,923,0.0296775023311
DAI62779,DAI85309,918,0.0295167357963
ELE32164,GRO59710,911,0.0292916626475
DAI62779,DAI75645,882,0.0283592167454
FRO40251,GRO73461,882,0.0283592167454
DAI62779,ELE92920,877,0.0281984502106
FRO40251,FRO92469,835,0.026848011318
DAI62779,ELE32164,832,0.0267515513971
DAI75645,GRO73461,712,0.0228931545609
DAI43223,ELE32164,711,0.022861001254
DAI62779,GRO30386,709,0.02279669464
ELE17451,FRO40251,697,0.0224108549564
DAI85309,ELE99737,659,0.0211890292917
DAI62779,ELE26917,650,0.020899649529
GRO21487,GRO73461,631,0.0202887366966
DAI62779,SNA45677,604,0.0194205974084
ELE17451,SNA80324,597,0.0191955242597
DAI62779,GRO71621,595,0.0191312176457
DAI62779,SNA55762,593,0.0190669110318
DAI62779,DAI83733,586,0.018841837883
ELE17451,GRO73461,580,0.0186489180412
GRO73461,SNA80324,562,0.0180701585158
DAI62779,GRO59710,561,0.0180380052088
DAI62779,FRO80039,550,0.0176843188322
DAI75645,ELE17451,547,0.0175878589113
DAI62779,SNA93860,537,0.0172663258416
DAI55148,DAI62779,526,0.016912639465
DAI43223,GRO59710,512,0.0164624931674
ELE17451,ELE32164,511,0.0164303398605
DAI62779,SNA18336,506,0.0162695733256
ELE32164,GRO73461,486,0.0156265071863
DAI62779,FRO78087,482,0.0154978939584
DAI85309,ELE17451,482,0.0154978939584
DAI62779,GRO94758,479,0.0154014340375
DAI62779,GRO21487,471,0.0151442075817
GRO85051,SNA80324,471,0.0151442075817
ELE17451,GRO30386,468,0.0150477476608
FRO85978,SNA95666,463,0.014886981126
DAI62779,FRO19221,462,0.014854827819
DAI62779,GRO46854,461,0.0148226745121
DAI43223,DAI62779,459,0.0147583678981
ELE92920,SNA18336,455,0.0146297546703
DAI88079,FRO40251,446,0.0143403749076
```

I ran both jobs on a Softlayer Hadoop 2.7.2 cluster with 1 master and 2 slave VMs. Each VM is running CentOS7.0-64 on 2 2.0 GHz cores with 6GB of RAM.

In each map reduce jobs, there were two mapper and one reducer tasks. The combiner was called twice.

According to the output, the entire MapReduce job ran in 44 seconds. The ,apper part ran in 36 seconds while the reducer part ran in 8 seconds. The second map reduce job ran in 34 seconds.

1.7 HW3.5: Stripes

Repeat 3.4 using the stripes design pattern for finding cooccuring pairs.

Report the compute times for stripes job versus the Pairs job. Describe the computational setup used (E.g., single computer; dual core; linux, number of mappers, number of reducers)

Instrument your mapper, combiner, and reducer to count how many times each is called using Counters and report these counts. Discuss the differences in these counts between the Pairs and Stripes jobs

In [149]:
```
%%writefile mapper.py
#!/usr/bin/python

#HW 3.4 - Mapper Function Code
from __future__ import print_function
from sets import Set

import sys,re
sys.stderr.write("reporter:counter:3.5,MapperCount,1\n")

#Define data split for custom partitioner
baskets =0

for line in sys.stdin:
    line=line.strip()
    # Using a set to avoid double counting in instances where a customer bought more than 1 o
    basket = sorted(Set(re.findall("[\w']+",line)))
    baskets +=1
    for i,p1 in enumerate(basket[:-1]):
        stripe = dict([(x,1) for x in basket[i+1:]])
        print (p1,stripe,sep='\t')

#Output total number of carts with a special key for order-inversion purposes
print (" Total",baskets,sep='\t')
```

```
Overwriting mapper.py
```

In [150]:
```
%%writefile combiner.py
#!/usr/bin/python
# combiner.py
# Author:Jackson Lane
# Description: combiner code for HW3.5
from __future__ import print_function

import sys

sys.stderr.write("reporter:counter:3.5,Combiner,1\n")
product =''
stripe = {}
for line in sys.stdin:
    #Parse line into fields
    sys.stderr.write(line)

    line = line.strip()
    newproduct,newstripe=line.split("\t")
    if newproduct==" Total": #Extract total products for order inversion
        print (" Total",int(newstripe)  ,sep='\t')
        continue
    newstripe = eval(newstripe)

    if product == newproduct:
        stripe = { k: stripe.get(k, 0) + newstripe.get(k, 0) for k in set(stripe) | set(newst
```

```
            else:
                if len(stripe) > 0: print (product,stripe,sep='\t')
                product=newproduct
                stripe=newstripe

        if len(stripe) > 0: print (product,stripe,sep='\t')
```

```
Overwriting combiner.py
```

```
In [151]: %%writefile reducer.py
          #!/usr/bin/python
          # reducer.py
          # Author:Jackson Lane
          # Description: reducer code for HW3.5
          # Reads in stripes pattern from mapper and breaks up into product pairs

          from __future__ import print_function
          import sys

          sys.stderr.write("reporter:counter:3.5,ReducerCount,1\n")
          baskets = float(0)
          product =''
          stripe = {}
          for line in sys.stdin:
              #Parse line into fields
              sys.stderr.write(line)
              line = line.strip()
              newproduct,newstripe=line.split("\t")
              if newproduct=="Total": #Extract total products for order inversion
                  baskets+=int(newstripe)
                  continue
              newstripe = eval(newstripe)

              if product == newproduct:
                  stripe = { k: stripe.get(k, 0) + newstripe.get(k, 0) for k in set(stripe) | set(newst
              else:
                  for (product2,count) in stripe.iteritems():
                      count = int(count)
                      if (count > 100): print(product,product2,count,count / baskets, sep = ",")
                  product=newproduct
                  stripe=newstripe

          for (product2,count) in stripe.iteritems():
                      count = int(count)
                      pair = (product,product2)
                      if (count > 100): print(product,product2,count,count / baskets, sep= ",")
```

```
Overwriting reducer.py
```

Run the mapreduce job

```
In [152]: !hdfs dfs -rm -r temp

          !hdfs dfs -put -p -f ProductPurchaseData.txt
```

```
!hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
-D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCompara
-D mapreduce.partition.keycomparator.options='-k1,1' \
-D stream.num.map.output.key.fields=2 \
-D stream.num.reduce.output.key.fields=2 \
-file mapper.py \
-file combiner.py \
-file reducer.py \
-mapper "mapper.py" \
-combiner "combiner.py" \
-reducer "reducer.py" \
-input ProductPurchaseData.txt \
-output temp

!hdfs dfs -rm -r results/3.5

# Second map reduce job to sort the output.
!hdfs dfs -put -p -f ProductPurchaseData.txt
!hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2*.jar \
-D mapreduce.job.output.key.comparator.class=org.apache.hadoop.mapred.lib.KeyFieldBasedCompara
-D mapreduce.partition.keycomparator.options='-k3,3nr -k1,2' \
-D mapreduce.output.key.field.separator="," \
-D stream.map.output.field.separator=, \
-D stream.reduce.output.field.separator=, \
-D stream.map.input.field.separator=, \
-D stream.reduce.input.field.separator=, \
-D map.output.key.field.separator=, \
-D stream.num.map.output.key.fields=3 \
-D stream.num.reduce.output.key.fields=3 \
-mapper cat \
-reducer cat \
-input temp/part-* \
-output results/3.5
```

```
16/06/05 13:43:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:43:40 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minute
Deleted temp
16/06/05 13:43:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:43:43 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files
16/06/05 13:43:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [mapper.py, combiner.py, reducer.py, /tmp/hadoop-unjar3031916136637617 40/] [] /tmp/strea
16/06/05 13:43:44 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:43:45 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:43:45 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 13:43:45 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 13:43:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0242
16/06/05 13:43:46 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0242
16/06/05 13:43:46 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicati
16/06/05 13:43:46 INFO mapreduce.Job: Running job: job_1464324416493_0242
16/06/05 13:43:53 INFO mapreduce.Job: Job job_1464324416493_0242 running in uber mode : false
16/06/05 13:43:53 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 13:44:05 INFO mapreduce.Job:  map 67% reduce 0%
16/06/05 13:45:08 INFO mapreduce.Job:  map 83% reduce 0%
16/06/05 13:45:09 INFO mapreduce.Job:  map 100% reduce 0%
```

```
16/06/05 13:45:19 INFO mapreduce.Job:  map 100% reduce 99%
16/06/05 13:45:20 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 13:45:21 INFO mapreduce.Job: Job job_1464324416493_0242 completed successfully
16/06/05 13:45:21 INFO mapreduce.Job: Counters: 52
        File System Counters
                FILE: Number of bytes read=15696092
                FILE: Number of bytes written=31758558
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3462815
                HDFS: Number of bytes written=52179
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=590408
                Total time spent by all reduces in occupied slots (ms)=70120
                Total time spent by all map tasks (ms)=147602
                Total time spent by all reduce tasks (ms)=8765
                Total vcore-milliseconds taken by all map tasks=147602
                Total vcore-milliseconds taken by all reduce tasks=8765
                Total megabyte-milliseconds taken by all map tasks=604577792
                Total megabyte-milliseconds taken by all reduce tasks=71802880
        Map-Reduce Framework
                Map input records=31101
                Map output records=349722
                Map output bytes=42019242
                Map output materialized bytes=15696098
                Input split bytes=202
                Combine input records=349722
                Combine output records=16942
                Reduce input groups=16930
                Reduce shuffle bytes=15696098
                Reduce input records=16942
                Reduce output records=1311
                Spilled Records=33884
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=459
                CPU time spent (ms)=138410
                Physical memory (bytes) snapshot=840298496
                Virtual memory (bytes) snapshot=19004043264
                Total committed heap usage (bytes)=694681600
        3.5
                Combiner=2
                MapperCount=2
                ReducerCount=1
        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=3462613
        File Output Format Counters
                Bytes Written=52179
16/06/05 13:45:21 INFO streaming.StreamJob: Output directory: temp
16/06/05 13:45:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:45:22 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minut
Deleted results/3.5
16/06/05 13:45:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
16/06/05 13:45:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
packageJobJar: [/tmp/hadoop-unjar2395628589614226864/] [] /tmp/streamjob8858589019399778405.jar tmpDir=
16/06/05 13:45:27 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:45:27 INFO client.RMProxy: Connecting to ResourceManager at /50.23.93.133:8032
16/06/05 13:45:28 INFO mapred.FileInputFormat: Total input paths to process : 1
16/06/05 13:45:28 INFO mapreduce.JobSubmitter: number of splits:2
16/06/05 13:45:28 INFO Configuration.deprecation: map.output.key.field.separator is deprecated. Instead
16/06/05 13:45:28 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1464324416493_0243
16/06/05 13:45:28 INFO impl.YarnClientImpl: Submitted application application_1464324416493_0243
16/06/05 13:45:28 INFO mapreduce.Job: The url to track the job: http://50.23.93.133:8088/proxy/applicat
16/06/05 13:45:28 INFO mapreduce.Job: Running job: job_1464324416493_0243
16/06/05 13:45:36 INFO mapreduce.Job: Job job_1464324416493_0243 running in uber mode : false
16/06/05 13:45:36 INFO mapreduce.Job:  map 0% reduce 0%
16/06/05 13:45:43 INFO mapreduce.Job:  map 100% reduce 0%
16/06/05 13:45:48 INFO mapreduce.Job:  map 100% reduce 100%
16/06/05 13:45:49 INFO mapreduce.Job: Job job_1464324416493_0243 completed successfully
16/06/05 13:45:49 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=54807
                FILE: Number of bytes written=472523
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=56461
                HDFS: Number of bytes written=52179
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=37796
                Total time spent by all reduces in occupied slots (ms)=26816
                Total time spent by all map tasks (ms)=9449
                Total time spent by all reduce tasks (ms)=3352
                Total vcore-milliseconds taken by all map tasks=9449
                Total vcore-milliseconds taken by all reduce tasks=3352
                Total megabyte-milliseconds taken by all map tasks=38703104
                Total megabyte-milliseconds taken by all reduce tasks=27459584
```

```
        Map-Reduce Framework
                Map input records=1311
                Map output records=1311
                Map output bytes=52179
                Map output materialized bytes=54813
                Input split bytes=186
                Combine input records=0
                Combine output records=0
                Reduce input groups=1311
                Reduce shuffle bytes=54813
                Reduce input records=1311
                Reduce output records=1311
                Spilled Records=2622
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=285
                CPU time spent (ms)=2250
                Physical memory (bytes) snapshot=692301824
                Virtual memory (bytes) snapshot=18996404224
                Total committed heap usage (bytes)=623378432
        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=56275
        File Output Format Counters
                Bytes Written=52179
16/06/05 13:45:49 INFO streaming.StreamJob: Output directory: results/3.5
```

```
In [153]: !hdfs dfs -cat results/3.4/part-00000 | head -50
```

```
16/06/05 13:45:50 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
DAI62779,ELE17451,1592,0.0511880646925
FRO40251,SNA80324,1412,0.0454004694383
DAI75645,FRO40251,1254,0.0403202469374
FRO40251,GRO85051,1213,0.0390019613517
DAI62779,GRO73461,1139,0.0366226166361
DAI75645,SNA80324,1130,0.0363332368734
DAI62779,FRO40251,1070,0.0344040384554
DAI62779,SNA80324,923,0.0296775023311
DAI62779,DAI85309,918,0.0295167357963
ELE32164,GRO59710,911,0.0292916626475
DAI62779,DAI75645,882,0.0283592167454
FRO40251,GRO73461,882,0.0283592167454
DAI62779,ELE92920,877,0.0281984502106
FRO40251,FRO92469,835,0.026848011318
DAI62779,ELE32164,832,0.0267515513971
DAI75645,GRO73461,712,0.0228931545609
DAI43223,ELE32164,711,0.022861001254
DAI62779,GRO30386,709,0.02279669464
```

```
ELE17451,FRO40251,697,0.0224108549564
DAI85309,ELE99737,659,0.0211890292917
DAI62779,ELE26917,650,0.020899649529
GRO21487,GRO73461,631,0.0202887366966
DAI62779,SNA45677,604,0.0194205974084
ELE17451,SNA80324,597,0.0191955242597
DAI62779,GRO71621,595,0.0191312176457
DAI62779,SNA55762,593,0.0190669110318
DAI62779,DAI83733,586,0.018841837883
ELE17451,GRO73461,580,0.0186489180412
GRO73461,SNA80324,562,0.0180701585158
DAI62779,GRO59710,561,0.0180380052088
DAI62779,FRO80039,550,0.0176843188322
DAI75645,ELE17451,547,0.0175878589113
DAI62779,SNA93860,537,0.0172663258416
DAI55148,DAI62779,526,0.016912639465
DAI43223,GRO59710,512,0.0164624931674
ELE17451,ELE32164,511,0.0164303398605
DAI62779,SNA18336,506,0.0162695733256
ELE32164,GRO73461,486,0.0156265071863
DAI62779,FRO78087,482,0.0154978939584
DAI85309,ELE17451,482,0.0154978939584
DAI62779,GRO94758,479,0.0154014340375
DAI62779,GRO21487,471,0.0151442075817
GRO85051,SNA80324,471,0.0151442075817
ELE17451,GRO30386,468,0.0150477476608
FRO85978,SNA95666,463,0.014886981126
DAI62779,FRO19221,462,0.014854827819
DAI62779,GRO46854,461,0.0148226745121
DAI43223,DAI62779,459,0.0147583678981
ELE92920,SNA18336,455,0.0146297546703
DAI88079,FRO40251,446,0.0143403749076
```

I ran both jobs on a Softlayer Hadoop 2.7.2 cluster with 1 master and 2 slave VMs. Each VM is running CentOS7.0-64 on 2 2.0 GHz cores with 6GB of RAM.

In each map reduce jobs, there were two mapper and one reducer tasks, and the counters reflected this. The combiner was called twice.

According to the output, the entire MapReduce job ran in 101 seconds. The Mapper part ran in 93 seconds while the reducer part ran in 8 seconds. The second map reduce job ran in 26 seconds.

The stripes job definetely ran slower than pairs job. I believe this is because the Mapper processed the stripe as a key, and that processing slowed the mapper down signficantly.

In [ ]: