Homework #2

Student: Alex Smith
Course: W261 - Machine Learning at Scale
Professor: Jimi Shanahan
Due Date: May 31

Useful resources

When completing this notebook, I found the following resources particularly useful:

- Week 3 Slides (https://www.dropbox.com/s/jwycz91sdi549ih/MIDS-LSML-2016-Lecture03-Map%20Reduce%20Algorithm%20Design-LiveSession3-2016-05-24.pdf?dl=0)
- Python Sorting Dictionaries (http://stackoverflow.com/questions/7742752/sorting-a-dictionary-by-value-then-by-key)
- Async Lecture on Pairs & Stripes (3.9)

Libraries

The following libraries must be installed before running the below code. They can all be installed through Pip (https://github.com/pypa/pip).

- Scikit Learn (http://scikit-learn.org/stable/)
- Numpy (http://www.numpy.org/)
- Regular Expression (https://docs.python.org/2/library/re.html)
- Pretty Table (https://pypi.python.org/pypi/PrettyTable)
- Random (https://docs.python.org/2/library/random.html)
- Datetime (https://docs.python.org/2/library/datetime.html)

Environment: Cloudera

This notebook is designed to run in a Cloudera virtual box. To set up a virtual box like this one, follow the instructions here (https://docs.google.com/presentation/d/1qCQM-2U2C6e584uM9kqTGr675K3_a8M1mEZaiT4Wmi8/edit#slide=id.p). Before beginning, make sure that you have started (in the following order) from the Cloudera manager:

1. Zookeeper
2. Yarn
3. HDFS

Setting up our Hadoop file system

In [3]:
```
!hdfs dfs -mkdir -p /user/cloudera/w261
```

Setting up some testing files

After you have gotten your data files into data/, let's make some testing files. This is helpful for running quick tests without churning through the whole data.

In [256]:
```
!head -50 data/Consumer_Complaints.csv > data/Consumer_test.csv
!head -50 data/ProductPurchaseData.txt > data/ProductPurchaseData_test.t
xt
```

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]
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. What is the Hadoop shuffle?

After our mappers finish with the data, we feed the data into our reducers. We want to make sure that all of the values for each key are given to a single reducer. For example, in a word count program, we don't want two reducers summing values for a single word. During this phase, we can merge the two sorted lists/arrays of the form [key,value] before feeding them into the reducers. This means that we have multiple lines for a certain [key,value] pair, we can merge them by key before passing them into the reducer. This is useful because it reduces the volume of data we are moving around. We always want to reduce the volume of this data because it is risky to move data around.
To help us reduce the amount of data we move around, we can employ a combiner. The combiner is what helps merge the [key,value] pairs by the key. It is done on the mapper node before transferring the data to the reducer. We can also use a combiner on the reducer note to help simplify the data before it enters the reducer program. We can have multiple combiners at multiple points that help consolidate our data.
For example, we might want to use a combiner in the classic word count problem. We might want to combine the mapper results from the multiple mappers on a single mapper node. If we have 2 mappers running on a single, they will both produce a list of words (the key) and a vale of 1. We can combine the outputs of the two mappers by summing the values between the outputs. For instance, if mapper 1 outputs "pig 1" and mapper 2 outputs "pig 1", we can send to the reducer "pig 2". This turned two lines of data into 1.
The Hadoop **shuffle** is all the steps between mapper output and reducer input. It is helpful to break it down into five simple steps:

1. Partition, sort, combine: for a given mapper, let's sort the output and merge the [key,value] pairs by key
2. Mergesort: for all the mappers on a given node, let's sort the outputs and merge the [key,value] pairs by key
3. Send to reducer: send the data from each mapper node to the reducer
4. Mergesort again: for all the outputs from each of the mapper nodes, do a another mergesort to merge the [key,value] pairs by key
5. Stream to reducer: finally send the data to the reducer program

HW3.1 consumer complaints dataset: 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 (https://www.dropbox.com/s/vbalm3yva2rr86m/Consumer_Complaints.csv?dl=0) provide here to complete this question.
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?

Here's is the first few lines of the of the Consumer Complaints Dataset:
1114245,Debt collection,Medical,Disclosure verification of debt,Not given enough info to verify debt,FL,32219,Web,11/13/2014,11/13/2014,"Choice Recovery, Inc.",Closed with explanation,Yes,
1114488,Debt collection,Medical,Disclosure verification of debt,Right to dispute notice not received,TX,75006,Web,11/13/2014,11/13/2014,"Expert Global Solutions, Inc.",In progress,Yes,
1114255,Bank account or service,Checking account,Deposits and withdrawals,,NY,11102,Web,11/13/2014,11/13/2014,"FNIS (Fidelity National Information Services, Inc.)",In progress,Yes,
1115106,Debt collection,"Other (phone, health club, etc.)",Communication tactics,Frequent or repeated calls,GA,31721,Web,11/13/2014,11/13/2014,"Expert Global Solutions, Inc.",In progress,Yes,

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.

Mapper function

We write a mapper function that takes an input of consumer complaints and outputs the category (Debt collection, Mortgage, or Other) with the number 1. This is the [key,value] pair. The mapper also writes to Hadoop counters for each of these categories.

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

# import the system library to read from
# the input and also write to stderror
import sys

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by commas since
    # this file is a CSV
    line = line.strip().split(",")

    # set the category as the second
    # item in the line
    category = line[1]

    # check which of the 3 categoreies
    # the line falls into and print that
    # with a 1 to the standard output
    # also print the counter to the
    # standard error so that we can double
    # check our results with the counter
    if category == "Debt collection":
        print "Debt collection\t1"
        sys.stderr.write("reporter:counter:Complaints,Debt collection,1
\n")
    elif category == "Mortgage":
        print "Mortgage\t1"
        sys.stderr.write("reporter:counter:Complaints,Mortgage,1\n")
    else:
        print "Other\t1"
        sys.stderr.write("reporter:counter:Complaints,Other,1\n")
```

```
Writing mapper.py
```

Reducer function

We write a reducer function to sum across the categories, and output the number of records in each of the three categores: debt collection, mortage, and other.

In [6]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.1

# import the system library to read from
# the input and also write to the stderror
import sys

# create a dictionary to store the categories
# and their respective counts
categories = {}

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by tabs
    line = line.strip().split("\t")

    # set the category as the first
    # item in the line
    category = line[0]

    # set the count as the second
    # item in the line
    count = int(line[1])

    # check to see if the category already
    # exists in the dictionary
    if category not in categories:

        # if it's not, add it, and initalize
        # it with a count of 0
        categories[category] = 0

    # increment the count
    categories[category] = categories[category] +\
    count

# print the outputs
for category in categories.keys():
    print category,"\t",categories[category]
```

```
Overwriting reducer.py
```

Make the files executable and run the MapReduce job in Hadoop

In [7]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_Complaints.cs
v /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-1
!rm -r /home/cloudera/w261/Outputs/Out_3_1

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-file /home/cloudera/w261/mapper.py    -mapper /home/cloudera/w261/mappe
r.py \
-file /home/cloudera/w261/reducer.py   -reducer /home/cloudera/w261/redu
cer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-1

# copy the output file to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-1/part-00000 /home/c
loudera/w261/Outputs/

# rename the file
!mv /home/cloudera/w261/Outputs/part-00000 /home/cloudera/w261/Outputs/O
ut_3_1
```

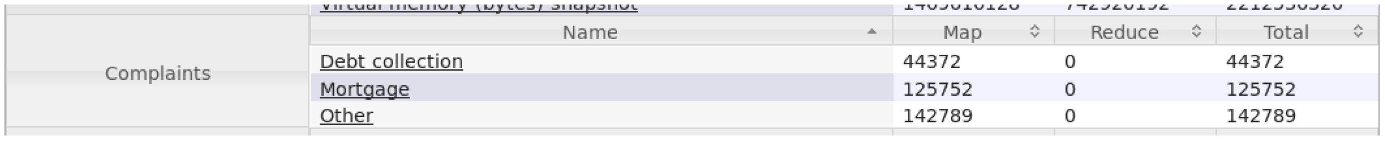
```
16/05/29 17:23:40 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_Complaints.csv' to trash a
t: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cl
oudera/w261/Consumer_Complaints.csv1464567820379
16/05/29 17:23:48 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-1' to trash at: hdfs://quic
kstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-ou
tput-3-11464567828574
rm: cannot remove `/home/cloudera/w261/Outputs/Out_3_1': No such file o
r directory
16/05/29 17:23:50 WARN streaming.StreamJob: -file option is deprecated,
 please use generic option -files instead.
packageJobJar: [/home/cloudera/w261/mapper.py, /home/cloudera/w261/redu
cer.py] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/streamjob5
786349409546119012.jar tmpDir=null
16/05/29 17:23:51 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/05/29 17:23:52 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/05/29 17:23:53 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/05/29 17:23:53 INFO mapreduce.JobSubmitter: number of splits:2
16/05/29 17:23:53 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464566418241_0003
16/05/29 17:23:54 INFO impl.YarnClientImpl: Submitted application appli
cation_1464566418241_0003
16/05/29 17:23:54 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464566418241_0003/
16/05/29 17:23:54 INFO mapreduce.Job: Running job: job_1464566418241_00
03
16/05/29 17:24:02 INFO mapreduce.Job: Job job_1464566418241_0003 runnin
g in uber mode : false
16/05/29 17:24:02 INFO mapreduce.Job:  map 0% reduce 0%
16/05/29 17:24:20 INFO mapreduce.Job:  map 67% reduce 0%
16/05/29 17:24:21 INFO mapreduce.Job:  map 100% reduce 0%
16/05/29 17:24:34 INFO mapreduce.Job:  map 100% reduce 100%
16/05/29 17:24:35 INFO mapreduce.Job: Job job_1464566418241_0003 comple
ted successfully
16/05/29 17:24:35 INFO mapreduce.Job: Counters: 52
        File System Counters
                FILE: Number of bytes read=187260
                FILE: Number of bytes written=736181
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50946999
                HDFS: Number of bytes written=54
                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)=407
0528
                Total time spent by all reduces in occupied slots (ms)=
1299712
                Total time spent by all map tasks (ms)=31801
                Total time spent by all reduce tasks (ms)=10154
                Total vcore-seconds taken by all map tasks=31801
                Total vcore-seconds taken by all reduce tasks=10154
                Total megabyte-seconds taken by all map tasks=4070528
                Total megabyte-seconds taken by all reduce tasks=129971
2
        Map-Reduce Framework
                Map input records=312913
                Map output records=312913
                Map output bytes=3324280
                Map output materialized bytes=187302
                Input split bytes=252
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=187302
                Reduce input records=312913
                Reduce output records=3
                Spilled Records=625826
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=390
                CPU time spent (ms)=12030
                Physical memory (bytes) snapshot=390012928
                Virtual memory (bytes) snapshot=2212536320
                Total committed heap usage (bytes)=147324928
        Complaints
                Debt collection=44372
                Mortgage=125752
                Other=142789
        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=50946747
        File Output Format Counters
                Bytes Written=54
16/05/29 17:24:35 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-1
```

Print the output to the notebook

In [9]:
```
!cat Outputs/Out_3_1 | sort -k1,1
```

```
Debt collection         44372
Mortgage        125752
Other   142789
```

We can compare the output from the MapReduce function with what the counters calculated. We see that we get the same result!



	Name	Map	Reduce	Total
Complaints	Debt collection	44372	0	44372
	Mortgage	125752	0	125752
	Other	142789	0	142789

HW 3.2 (a) Analyze the performance of your Mappers, Combiners and Reducers using Counters

For this brief study the Input file will be one record (the next line only):
foo foo quux labs foo bar quux

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.
Note: we split the instructions between different cells because each instruction is relatively self-contained.

Write the line to a file

In [13]:
```
!echo "foo foo quux labs foo bar quux" > data/Input_3_2_a
```

Mapper function

This function takes an input, splits the line into words by white space, and outputs a key,value pair where the key is the word and the value is 1.

In [15]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.2(a)

# import the system library to read from
# the input and also write to stderr
import sys

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Mapper,1\n")

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by spaces
    line = line.strip().split()

    # print out the key-value pair as a tab
    # delimited list of word and 1
    for word in line:
        print word,"\t",1
```

```
Overwriting mapper.py
```

Reducer function

This function merges the counts for each word and outputs a list of words with their associated counts.

In [16]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.2(a)

# import the system library to read from
# the input and also write to the stderror
import sys

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Reducer,1\n")

# create a dictionary to store the word
# counts
words = {}

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by tabs
    line = line.strip().split("\t")

    # set the category as the first
    # item in the line
    word = line[0]

    # set the count as the second
    # item in the line
    count = int(line[1])

    # check to see if the word already
    # exists in the dictionary
    if word not in words:

        # if it's not, add it, and initalize
        # it with a count of 0
        words[word] = 0

    # increment the count
    words[word] = words[word] + count

# print the outputs
for word in words.keys():
    print word,"\t",words[word]
```

```
Overwriting reducer.py
```

Make the files executable and run the MapReduce job in Hadoop

In [47]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Input_3_2_a /user/clou
dera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-a
!rm -r /home/cloudera/w261/Outputs/Out_3_2-a

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-a

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=1 -D mapred.reduce.tasks=4 \
-file /home/cloudera/w261/mapper.py    -mapper /home/cloudera/w261/mappe
r.py \
-file /home/cloudera/w261/reducer.py   -reducer /home/cloudera/w261/redu
cer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-a

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-a/* /home/clouder
a/w261/Outputs/Out_3_2-a
```

```
16/05/29 19:34:20 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Input_3_2_a' to trash at: hdfs://qui
ckstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261/I
nput_3_2_a1464575660216
16/05/29 19:34:26 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-2-a' to trash at: hdfs://qu
ickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-
output-3-2-a1464575666879
16/05/29 19:34:28 WARN streaming.StreamJob: -file option is deprecated,
 please use generic option -files instead.
packageJobJar: [/home/cloudera/w261/mapper.py, /home/cloudera/w261/redu
cer.py] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/streamjob4
194598760713788390.jar tmpDir=null
16/05/29 19:34:30 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/05/29 19:34:30 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/05/29 19:34:31 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/05/29 19:34:31 INFO mapreduce.JobSubmitter: number of splits:1
16/05/29 19:34:31 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
16/05/29 19:34:31 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/05/29 19:34:31 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464566418241_0007
16/05/29 19:34:32 INFO impl.YarnClientImpl: Submitted application appli
cation_1464566418241_0007
16/05/29 19:34:32 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464566418241_0007/
16/05/29 19:34:32 INFO mapreduce.Job: Running job: job_1464566418241_00
07
16/05/29 19:34:42 INFO mapreduce.Job: Job job_1464566418241_0007 runnin
g in uber mode : false
16/05/29 19:34:42 INFO mapreduce.Job:  map 0% reduce 0%
16/05/29 19:34:57 INFO mapreduce.Job:  map 100% reduce 0%
16/05/29 19:35:18 INFO mapreduce.Job:  map 100% reduce 50%
16/05/29 19:35:32 INFO mapreduce.Job:  map 100% reduce 100%
16/05/29 19:35:33 INFO mapreduce.Job: Job job_1464566418241_0007 comple
ted successfully
16/05/29 19:35:33 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=132
                FILE: Number of bytes written=602905
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=145
                HDFS: Number of bytes written=34
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=8
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=4
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=168
```

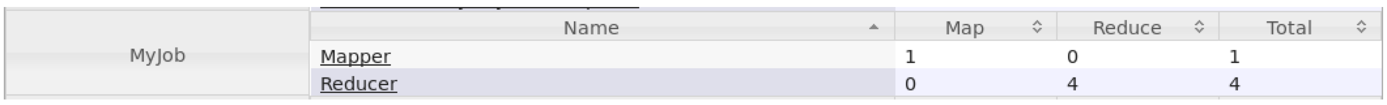
```
4864
                Total time spent by all reduces in occupied slots (ms)=
7617536
                Total time spent by all map tasks (ms)=13163
                Total time spent by all reduce tasks (ms)=59512
                Total vcore-seconds taken by all map tasks=13163
                Total vcore-seconds taken by all reduce tasks=59512
                Total megabyte-seconds taken by all map tasks=1684864
                Total megabyte-seconds taken by all reduce tasks=761753
6
        Map-Reduce Framework
                Map input records=1
                Map output records=7
                Map output bytes=52
                Map output materialized bytes=116
                Input split bytes=114
                Combine input records=0
                Combine output records=0
                Reduce input groups=4
                Reduce shuffle bytes=116
                Reduce input records=7
                Reduce output records=4
                Spilled Records=14
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=639
                CPU time spent (ms)=7150
                Physical memory (bytes) snapshot=582488064
                Virtual memory (bytes) snapshot=3701907456
                Total committed heap usage (bytes)=232259584
        MyJob
                Mapper=1
                Reducer=4
        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=31
        File Output Format Counters
                Bytes Written=34
16/05/29 19:35:33 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-a
```

Print the output

In [48]:
```
!cat Outputs/Out_3_2-a/* | sort -n -r -k2
```
```
foo     3
quux    2
labs    1
bar     1
```

Number of counters

We use a single mapper and 4 reducers. In this simple problem, we use a single mapper because our input file is small and does not need to be chunked. We didn't need to use 4 reducers, but we go ahead with 4 so that each reducer handles 1 word and only 1 word. We might want to segregate each reducer to a single word if we were analyzing a very large corpus. This would be an intuitive way to break up the output files.



	Name	Map	Reduce	Total
MyJob	Mapper	1	0	1
	Reducer	0	4	4

HW 3.2 (b)

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 times 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?

Mapper function

This function takes the string from the issue column and outputs a line for each word following by the integer 1.

In [259]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.2(b)

# import the system library to read from
# the input and also write to stderr
import sys

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Mapper,1\n")

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by spaces
    line = line.strip().split(",")

    # grab the words we're interested in
    words = line[3].split()

    # print out the key-value pair as a tab
    # delimited list of word and 1
    for word in words:
        print word,"\t",1
```

```
Overwriting mapper.py
```

Reducer function

This function takes the inputs of key,value pairs of words and counts and merges the counts by word to generate a list of total counts for each word.

In [260]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.2(b)

# import the system library to read from
# the input and also write to the stderror
import sys

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Reducer,1\n")

# create a dictionary to store the word
# counts
words = {}

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by tabs
    line = line.strip().split("\t")

    # set the category as the first
    # item in the line
    word = line[0]

    # set the count as the second
    # item in the line
    count = int(line[1])

    # check to see if the word already
    # exists in the dictionary
    if word not in words:

        # if it's not, add it, and initalize
        # it with a count of 0
        words[word] = 0

    # increment the count
    words[word] = words[word] + count

# print the outputs
for word in words.keys():
    print word,"\t",words[word]
```

```
Overwriting reducer.py
```

Make the files executable and run the test data in Hadoop

In [10]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_test.csv /use
r/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-b
!rm -r /home/cloudera/w261/Outputs/Out_3_2-b

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-b

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=2 \
-file /home/cloudera/w261/mapper.py    -mapper /home/cloudera/w261/mappe
r.py \
-file /home/cloudera/w261/reducer.py   -reducer /home/cloudera/w261/redu
cer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-b

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-b/* /home/clouder
a/w261/Outputs/Out_3_2-b
```

```
16/05/30 12:22:26 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_test.csv' to trash at: hdf
s://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/clouder
a/w261/Consumer_test.csv
16/05/30 12:22:40 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-2-b' to trash at: hdfs://qu
ickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-
output-3-2-b
16/05/30 12:22:42 WARN streaming.StreamJob: -file option is deprecated,
 please use generic option -files instead.
packageJobJar: [/home/cloudera/w261/mapper.py, /home/cloudera/w261/redu
cer.py] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/streamjob3
598338797964190338.jar tmpDir=null
16/05/30 12:22:45 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 12:22:45 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 12:22:47 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/05/30 12:22:48 INFO mapreduce.JobSubmitter: number of splits:2
16/05/30 12:22:48 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
16/05/30 12:22:48 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/05/30 12:22:49 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0002
16/05/30 12:22:49 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0002
16/05/30 12:22:50 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0002/
16/05/30 12:22:50 INFO mapreduce.Job: Running job: job_1464634906532_00
02
16/05/30 12:23:17 INFO mapreduce.Job: Job job_1464634906532_0002 runnin
g in uber mode : false
16/05/30 12:23:17 INFO mapreduce.Job:  map 0% reduce 0%
16/05/30 12:24:12 INFO mapreduce.Job:  map 100% reduce 0%
16/05/30 12:24:33 INFO mapreduce.Job:  map 100% reduce 100%
16/05/30 12:24:36 INFO mapreduce.Job: Job job_1464634906532_0002 comple
ted successfully
16/05/30 12:24:36 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=1768
                FILE: Number of bytes written=486360
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=61947
                HDFS: Number of bytes written=1195
                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=2
                Total time spent by all maps in occupied slots (ms)=136
32256
```

```
		Total time spent by all reduces in occupied slots (ms)=
4535424
		Total time spent by all map tasks (ms)=106502
		Total time spent by all reduce tasks (ms)=35433
		Total vcore-seconds taken by all map tasks=106502
		Total vcore-seconds taken by all reduce tasks=35433
		Total megabyte-seconds taken by all map tasks=13632256
		Total megabyte-seconds taken by all reduce tasks=453542
4
	Map-Reduce Framework
		Map input records=249
		Map output records=1083
		Map output bytes=11212
		Map output materialized bytes=2410
		Input split bytes=240
		Combine input records=0
		Combine output records=0
		Reduce input groups=102
		Reduce shuffle bytes=2410
		Reduce input records=1083
		Reduce output records=102
		Spilled Records=2166
		Shuffled Maps =4
		Failed Shuffles=0
		Merged Map outputs=4
		GC time elapsed (ms)=2022
		CPU time spent (ms)=5850
		Physical memory (bytes) snapshot=491601920
		Virtual memory (bytes) snapshot=2949853184
		Total committed heap usage (bytes)=190316544
	MyJob
		Mapper=2
		Reducer=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=61707
	File Output Format Counters
		Bytes Written=1195
16/05/30 12:24:36 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-b
```

Run the full data through Hadoop

In [261]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_Complaints.cs
v /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-b
!rm -r /home/cloudera/w261/Outputs/Out_3_2-b

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-b

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=2 \
-file /home/cloudera/w261/mapper.py    -mapper /home/cloudera/w261/mappe
r.py \
-file /home/cloudera/w261/reducer.py   -reducer /home/cloudera/w261/redu
cer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-b

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-b/* /home/clouder
a/w261/Outputs/Out_3_2-b
```

```
16/06/04 09:56:24 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_Complaints.csv' to trash a
t: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cl
oudera/w261/Consumer_Complaints.csv
16/06/04 09:56:35 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-2-b' to trash at: hdfs://qu
ickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-
output-3-2-b1465059395441
16/06/04 09:56:37 WARN streaming.StreamJob: -file option is deprecated,
 please use generic option -files instead.
packageJobJar: [/home/cloudera/w261/mapper.py, /home/cloudera/w261/redu
cer.py] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/streamjob1
815764423982164672.jar tmpDir=null
16/06/04 09:56:39 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 09:56:40 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 09:56:42 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 09:56:42 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 09:56:42 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
16/06/04 09:56:42 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/06/04 09:56:43 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0019
16/06/04 09:56:44 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0019
16/06/04 09:56:44 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0019/
16/06/04 09:56:44 INFO mapreduce.Job: Running job: job_1464634906532_00
19
16/06/04 09:57:04 INFO mapreduce.Job: Job job_1464634906532_0019 runnin
g in uber mode : false
16/06/04 09:57:04 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 09:57:58 INFO mapreduce.Job:  map 64% reduce 0%
16/06/04 09:58:02 INFO mapreduce.Job:  map 67% reduce 0%
16/06/04 09:58:04 INFO mapreduce.Job:  map 83% reduce 0%
16/06/04 09:58:05 INFO mapreduce.Job:  map 92% reduce 0%
16/06/04 09:58:07 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 09:58:31 INFO mapreduce.Job:  map 100% reduce 84%
16/06/04 09:58:34 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 09:58:41 INFO mapreduce.Job: Job job_1464634906532_0019 comple
ted successfully
16/06/04 09:58:41 INFO mapreduce.Job: Counters: 51
	File System Counters
		FILE: Number of bytes read=916715
		FILE: Number of bytes written=1976599
		FILE: Number of read operations=0
		FILE: Number of large read operations=0
		FILE: Number of write operations=0
		HDFS: Number of bytes read=50946999
		HDFS: Number of bytes written=2659
		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=2
                Total time spent by all maps in occupied slots (ms)=139
36128
                Total time spent by all reduces in occupied slots (ms)=
7184640
                Total time spent by all map tasks (ms)=108876
                Total time spent by all reduce tasks (ms)=56130
                Total vcore-seconds taken by all map tasks=108876
                Total vcore-seconds taken by all reduce tasks=56130
                Total megabyte-seconds taken by all map tasks=13936128
                Total megabyte-seconds taken by all reduce tasks=718464
0
        Map-Reduce Framework
                Map input records=312913
                Map output records=978634
                Map output bytes=10407527
                Map output materialized bytes=590545
                Input split bytes=252
                Combine input records=0
                Combine output records=0
                Reduce input groups=182
                Reduce shuffle bytes=590545
                Reduce input records=978634
                Reduce output records=182
                Spilled Records=2483605
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=2924
                CPU time spent (ms)=20260
                Physical memory (bytes) snapshot=497647616
                Virtual memory (bytes) snapshot=2973028352
                Total committed heap usage (bytes)=188743680
        MyJob
                Mapper=2
                Reducer=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=50946747
        File Output Format Counters
                Bytes Written=2659
16/06/04 09:58:41 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-b
```

Sample the output file

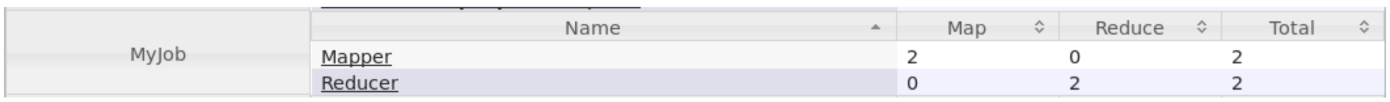
Let's print a sample from one of the output files

In [263]:
```
!head Outputs/Out_3_2-b/part-00000
```

```
"Account        16205
the     6248
increase/decrease       1149
APR     3431
Account         350
promised        274
Overlimit       127
process         5505
issue   1098
Debt    1343
```

Mapper and reducer counters

We used a total of 2 reducers and 2 mappers. We can use 2 reducers easily because we are not looking at connections between words, only the count of each word. Because Hadoop sorts by keys, each reducer will get all the records for a single word.



	Name	Map	Reduce	Total
MyJob	Mapper	2	0	2
	Reducer	0	2	2

3.2 (c)

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.

Mapper function

This function takes the words in the issue column and outputs each word with an integer 1.

In [264]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.2(c)

# import the system library to read from
# the input and also write to stderr
import sys

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Mapper,1\n")

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by spaces
    line = line.strip().split(",")

    # grab the words we're interested in
    words = line[3].split()

    # print out the key-value pair as a tab
    # delimited list of word and 1
    for word in words:
        print word,"\t",1
```

```
Overwriting mapper.py
```

Reducer function

This function takes the words outputted from the mapper and merges based on the word to get a sum of counts for each word.

In [265]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.2(c)

# import the system library to read from
# the input and also write to the stderror
import sys

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Reducer,1\n")

# create a dictionary to store the word
# counts
words = {}

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by tabs
    line = line.strip().split("\t")

    # set the category as the first
    # item in the line
    word = line[0]

    # set the count as the second
    # item in the line
    count = int(line[1])

    # check to see if the word already
    # exists in the dictionary
    if word not in words:

        # if it's not, add it, and initalize
        # it with a count of 0
        words[word] = 0

    # increment the count
    words[word] = words[word] + count

# print the outputs
for word in words.keys():
    print word,"\t",words[word]
```

```
Overwriting reducer.py
```

Make executable and run the test data in hadoop

We add our reducer as a stand-alone combiner.

In [15]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_test.csv /use
r/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-c
!rm -r /home/cloudera/w261/Outputs/Out_3_2-c

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-c

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=2 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-combiner reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-c

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-c/* /home/clouder
a/w261/Outputs/Out_3_2-c
```

```
16/05/30 12:54:01 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_test.csv' to trash at: hdf
s://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/clouder
a/w261/Consumer_test.csv1464638041156
16/05/30 12:54:09 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-2-c' to trash at: hdfs://qu
ickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-
output-3-2-c
16/05/30 12:54:13 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/05/30 12:54:13 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob247530583643913124.jar tmpDir=null
16/05/30 12:54:14 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 12:54:14 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 12:54:16 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/05/30 12:54:16 INFO mapreduce.JobSubmitter: number of splits:2
16/05/30 12:54:17 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0004
16/05/30 12:54:17 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0004
16/05/30 12:54:17 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0004/
16/05/30 12:54:17 INFO mapreduce.Job: Running job: job_1464634906532_00
04
16/05/30 12:54:26 INFO mapreduce.Job: Job job_1464634906532_0004 runnin
g in uber mode : false
16/05/30 12:54:26 INFO mapreduce.Job:  map 0% reduce 0%
16/05/30 12:54:52 INFO mapreduce.Job:  map 100% reduce 0%
16/05/30 12:55:14 INFO mapreduce.Job:  map 100% reduce 100%
16/05/30 12:55:16 INFO mapreduce.Job: Job job_1464634906532_0004 comple
ted successfully
16/05/30 12:55:16 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=1476
                FILE: Number of bytes written=486790
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=61947
                HDFS: Number of bytes written=1297
                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=2
                Total time spent by all maps in occupied slots (ms)=595
6480
                Total time spent by all reduces in occupied slots (ms)=
4658816
                Total time spent by all map tasks (ms)=46535
                Total time spent by all reduce tasks (ms)=36397
                Total vcore-seconds taken by all map tasks=46535
                Total vcore-seconds taken by all reduce tasks=36397
                Total megabyte-seconds taken by all map tasks=5956480
                Total megabyte-seconds taken by all reduce tasks=465881
6
        Map-Reduce Framework
                Map input records=249
                Map output records=1083
                Map output bytes=11212
                Map output materialized bytes=1940
                Input split bytes=240
                Combine input records=1083
                Combine output records=159
                Reduce input groups=156
                Reduce shuffle bytes=1940
                Reduce input records=159
                Reduce output records=102
                Spilled Records=318
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=667
                CPU time spent (ms)=4750
                Physical memory (bytes) snapshot=488947712
                Virtual memory (bytes) snapshot=2959560704
                Total committed heap usage (bytes)=191889408
        MyJob
                Mapper=2
                Reducer=6
        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=61707
        File Output Format Counters
                Bytes Written=1297
16/05/30 12:55:16 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-c
```

Run the full data through Hadoop

In [266]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_Complaints.cs
v /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-c
!rm -r /home/cloudera/w261/Outputs/Out_3_2-c

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-c

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=2 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-combiner reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-c

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-c/* /home/clouder
a/w261/Outputs/Out_3_2-c
```

```
16/06/04 10:10:41 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_Complaints.csv' to trash a
t: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cl
oudera/w261/Consumer_Complaints.csv1465060241812
16/06/04 10:10:57 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-2-c' to trash at: hdfs://qu
ickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-
output-3-2-c
16/06/04 10:11:01 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/06/04 10:11:01 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob438746980786419372.jar tmpDir=null
16/06/04 10:11:02 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:11:02 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:11:04 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 10:11:04 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 10:11:04 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0020
16/06/04 10:11:05 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0020
16/06/04 10:11:05 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0020/
16/06/04 10:11:05 INFO mapreduce.Job: Running job: job_1464634906532_00
20
16/06/04 10:11:16 INFO mapreduce.Job: Job job_1464634906532_0020 runnin
g in uber mode : false
16/06/04 10:11:16 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 10:11:54 INFO mapreduce.Job:  map 67% reduce 0%
16/06/04 10:11:58 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 10:12:18 INFO mapreduce.Job:  map 100% reduce 50%
16/06/04 10:12:19 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 10:12:21 INFO mapreduce.Job: Job job_1464634906532_0020 comple
ted successfully
16/06/04 10:12:21 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=12550
                FILE: Number of bytes written=499602
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50946999
                HDFS: Number of bytes written=2841
                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=2
                Total time spent by all maps in occupied slots (ms)=100
56576
                Total time spent by all reduces in occupied slots (ms)=
4282624
                Total time spent by all map tasks (ms)=78567
                Total time spent by all reduce tasks (ms)=33458
                Total vcore-seconds taken by all map tasks=78567
                Total vcore-seconds taken by all reduce tasks=33458
                Total megabyte-seconds taken by all map tasks=10056576
                Total megabyte-seconds taken by all reduce tasks=428262
4
        Map-Reduce Framework
                Map input records=312913
                Map output records=978634
                Map output bytes=10407527
                Map output materialized bytes=6208
                Input split bytes=252
                Combine input records=978634
                Combine output records=508
                Reduce input groups=506
                Reduce shuffle bytes=6208
                Reduce input records=508
                Reduce output records=182
                Spilled Records=1369
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=1229
                CPU time spent (ms)=9250
                Physical memory (bytes) snapshot=485023744
                Virtual memory (bytes) snapshot=2953555968
                Total committed heap usage (bytes)=192937984
        MyJob
                Mapper=2
                Reducer=8
        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=50946747
        File Output Format Counters
                Bytes Written=2841
16/06/04 10:12:21 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-c
```

Mapper and reducer counters

We use a total of 8 counters, 6 reducer counters and 2 mapper counters. On the mapper side, we use 2 mapper counters and 4 reducer counters. On the reducer side, we use 2 reducer counters. We only have 2 mappers and 2 reducers but the counters show more because we are using the reducers as combiners. We are running the reducers as combiners twice for each mapper. In the mapper, this could be happening once in the circular buffer and once after all the output has been spilled to disk.

HW3.2 (d)

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).

Mapper function

This function thakes the words from the complaints data and outputs each word with an integer 1.

```
In [276]: %%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.2(d)

# import the system library to read from
# the input and also write to stderr
import sys

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Mapper,1\n")

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by spaces
    line = line.strip().split(",")

    # grab the words we're interested in
    words = line[3].split()

    # print out the key-value pair as a tab
    # delimited list of word and 1
    for word in words:
        print word,"\t",1
```

```
Overwriting mapper.py
```

Reducer function

This function merges the counts for each word, by word to produce a sum of word counts. It returns a sorted list by the word count.

In [277]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.2(d)

# import the system library to read from
# the input and also write to the stderror
# import the operator library to help sort
# the dictionary
import sys
import operator

# add a counter line for each time the
# mapper is called
sys.stderr.write("reporter:counter:MyJob,Reducer,1\n")

# create a dictionary to store the word
# counts
words = {}

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by tabs
    line = line.strip().split("\t")

    # set the word as the first
    # item in the line
    word = line[0].strip()

    # set the count as the second
    # item in the line
    count = int(line[1])

    # check to see if the word already
    # exists in the dictionary
    if word not in words:

        # if it's not, add it, and initalize
        # it with a count of 0
        words[word] = 0

    # increment the count
    words[word] = words[word] + count

# print the outputs
for word in sorted(words.items(),\
                   key=operator.itemgetter(1)):
    print word[0],"\t",word[1]
```

```
Overwriting reducer.py
```

Make the files executable and run the hadoop

In [20]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_test.csv /use
r/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-d
!rm -r /home/cloudera/w261/Outputs/Out_3_2-d

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-d

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-combiner reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-d

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-d/part-00000 /hom
e/cloudera/w261/Outputs/Out_3_2-d
```

```
16/05/30 13:28:01 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_test.csv' to trash at: hdf
s://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/clouder
a/w261/Consumer_test.csv1464640081215
rm: `/user/cloudera/w261-output-3-2-d': No such file or directory
rm: cannot remove `/home/cloudera/w261/Outputs/Out_3_2-d': No such file
 or directory
16/05/30 13:28:19 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/05/30 13:28:19 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob6504188710895385985.jar tmpDir=null
16/05/30 13:28:20 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 13:28:21 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 13:28:23 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/05/30 13:28:24 INFO mapreduce.JobSubmitter: number of splits:2
16/05/30 13:28:24 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0005
16/05/30 13:28:25 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0005
16/05/30 13:28:25 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0005/
16/05/30 13:28:25 INFO mapreduce.Job: Running job: job_1464634906532_00
05
16/05/30 13:28:47 INFO mapreduce.Job: Job job_1464634906532_0005 runnin
g in uber mode : false
16/05/30 13:28:47 INFO mapreduce.Job:  map 0% reduce 0%
16/05/30 13:29:44 INFO mapreduce.Job:  map 100% reduce 0%
16/05/30 13:29:55 INFO mapreduce.Job:  map 100% reduce 100%
16/05/30 13:29:56 INFO mapreduce.Job: Job job_1464634906532_0005 comple
ted successfully
16/05/30 13:29:56 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=1357
                FILE: Number of bytes written=365632
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=61947
                HDFS: Number of bytes written=1093
                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)=136
84992
                Total time spent by all reduces in occupied slots (ms)=
1063296
                Total time spent by all map tasks (ms)=106914
                Total time spent by all reduce tasks (ms)=8307
                Total vcore-seconds taken by all map tasks=106914
                Total vcore-seconds taken by all reduce tasks=8307
                Total megabyte-seconds taken by all map tasks=13684992
                Total megabyte-seconds taken by all reduce tasks=106329
6
        Map-Reduce Framework
                Map input records=249
                Map output records=1083
                Map output bytes=11212
                Map output materialized bytes=1750
                Input split bytes=240
                Combine input records=1083
                Combine output records=159
                Reduce input groups=159
                Reduce shuffle bytes=1750
                Reduce input records=159
                Reduce output records=102
                Spilled Records=318
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=537
                CPU time spent (ms)=3430
                Physical memory (bytes) snapshot=389562368
                Virtual memory (bytes) snapshot=2207375360
                Total committed heap usage (bytes)=143130624
        MyJob
                Mapper=2
                Reducer=3
        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=61707
        File Output Format Counters
                Bytes Written=1093
16/05/30 13:29:56 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-d
```

Run the full data through Hadoop

In [278]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_Complaints.cs
v /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-d
!rm -r /home/cloudera/w261/Outputs/Out_3_2-d

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-d

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-combiner reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-d

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-d/part-00000 /hom
e/cloudera/w261/Outputs/Out_3_2-d
```

```
16/06/04 10:22:53 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_Complaints.csv' to trash a
t: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cl
oudera/w261/Consumer_Complaints.csv1465060973395
16/06/04 10:23:02 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-2-d' to trash at: hdfs://qu
ickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-
output-3-2-d
16/06/04 10:23:05 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/06/04 10:23:05 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob7367082926943165624.jar tmpDir=null
16/06/04 10:23:06 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:23:06 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:23:08 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 10:23:08 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 10:23:08 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0021
16/06/04 10:23:09 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0021
16/06/04 10:23:09 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0021/
16/06/04 10:23:09 INFO mapreduce.Job: Running job: job_1464634906532_00
21
16/06/04 10:23:19 INFO mapreduce.Job: Job job_1464634906532_0021 runnin
g in uber mode : false
16/06/04 10:23:19 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 10:23:56 INFO mapreduce.Job:  map 67% reduce 0%
16/06/04 10:23:59 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 10:24:10 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 10:24:10 INFO mapreduce.Job: Job job_1464634906532_0021 comple
ted successfully
16/06/04 10:24:10 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=9242
                FILE: Number of bytes written=377430
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50946999
                HDFS: Number of bytes written=2477
                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)=944
5888
                Total time spent by all reduces in occupied slots (ms)=
1024768
```

```
                Total time spent by all map tasks (ms)=73796
                Total time spent by all reduce tasks (ms)=8006
                Total vcore-seconds taken by all map tasks=73796
                Total vcore-seconds taken by all reduce tasks=8006
                Total megabyte-seconds taken by all map tasks=9445888
                Total megabyte-seconds taken by all reduce tasks=102476
8
        Map-Reduce Framework
                Map input records=312913
                Map output records=978634
                Map output bytes=10407527
                Map output materialized bytes=5663
                Input split bytes=252
                Combine input records=978634
                Combine output records=508
                Reduce input groups=508
                Reduce shuffle bytes=5663
                Reduce input records=508
                Reduce output records=182
                Spilled Records=1369
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=899
                CPU time spent (ms)=7400
                Physical memory (bytes) snapshot=376905728
                Virtual memory (bytes) snapshot=2190626816
                Total committed heap usage (bytes)=145227776
        MyJob
                Mapper=2
                Reducer=4
        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=50946747
        File Output Format Counters
                Bytes Written=2477
16/06/04 10:24:10 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-d
```

Top 50 tokens

These 50 words appeared most frequently.

In [281]:
```
!tail -50 Outputs/Out_3_2-d/part-00000 | sort -k2nr,2 -k1
```

```
"Loan	107254
modification	70487
servicing	36767
credit	36126
report	30546
Incorrect	29069
information	29069
on	29069
or	22533
debt	17966
and	16448
"Account	16205
opening	16205
Credit	14768
club	12545
health	12545
/	12386
not	12353
loan	12237
attempts	11848
collect	11848
Cont'd	11848
owed	11848
of	10885
my	10731
Deposits	10555
withdrawals	10555
Problems	9484
"Application	8625
to	8401
Billing	8158
Other	7886
disputes	6938
Communication	6920
tactics	6920
reporting	6559
lease	6337
the	6248
being	5663
by	5663
caused	5663
funds	5663
low	5663
process	5505
Disclosure	5214
verification	5214
Managing	5006
company's	4858
investigation	4858
card	4405
```

Bottom 10 tokens

These 10 words appear least frequently.

In [285]:
```
!head Outputs/Out_3_2-d/part-00000 | sort -k2nr,2 -k1
```

```
credited        92
Payment         92
checks  75
Convenience     75
amt     71
day     71
wrong   71
disclosures     64
Incorrect/missing       64
Issue   1
```

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.

Mapper function

This function takes an input of words and ouputs each word with it's associated count.

In [286]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.2.1

# import the system library to read from
# the input and also write to stderr
import sys

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by spaces
    line = line.strip().split(",")

    # grab the words we're interested in
    words = line[3].split()

    # print out the key-value pair as a tab
    # delimited list of word and 1
    for word in words:
        print word,"\t",1
```

```
Overwriting mapper.py
```

Reducer function

This function sums across the counts for a single word and outputs a list of word with the final counts.

In [287]:
```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.2.1

# import the system library to read from
# the input and also write to the stderror
# import the operator library to help sort
# the dictionary
import sys
import operator

# create a dictionary to store the word
# counts
words = {}

# loop through each line
for line in sys.stdin:

    # stirp off any extra spaces and
    # split the line by tabs
    line = line.strip().split("\t")

    # set the word as the first
    # item in the line
    word = line[0].strip()

    # set the count as the second
    # item in the line
    count = int(line[1])

    # check to see if the word already
    # exists in the dictionary
    if word not in words:

        # if it's not, add it, and initalize
        # it with a count of 0
        words[word] = 0

    # increment the count
    words[word] = words[word] + count

# print the outputs
for word in sorted(words.items(),\
                   key=operator.itemgetter(1)):
    print word[0],"\t",word[1]
```

```
Overwriting reducer.py
```

Test the code on the command line

In [251]:
```
# make the files executable
!chmod +x mapper.py
!chmod +x reducer.py

# first try with just the reducer
!cat data/Consumer_test.csv | ~/w261/mapper.py > Outputs/testing.txt

# read the first couple lines of the testing file
!head Outputs/testing.txt
```

```
Disclosure      1
verification    1
of      1
debt    1
Disclosure      1
verification    1
of      1
debt    1
Deposits        1
and     1
```

In [252]:
```
# now let's test the output of the mapper into
# the reducer
!cat data/Consumer_test.csv | ~/w261/mapper.py | ~/w261/reducer.py > Out
puts/testing.txt

# read the first couple lines of the testing file
!tail Outputs/testing.txt
```

```
health  26
Cont'd  26
club    26
owed    26
debt    34
on      74
Incorrect       74
information     74
report  77
credit  97
```

Make the files executable and run the hadoop with 2 reducers

In [253]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_test.csv /use
r/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-1
!rm -r /home/cloudera/w261/Outputs/Out_3_2-1

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-1

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=2 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-combiner reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-1

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-1/* /home/clouder
a/w261/Outputs/Out_3_2-1
```

```
16/06/04 08:32:41 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/ProductPurchaseData_test.txt' to tra
sh at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/use
r/cloudera/w261/ProductPurchaseData_test.txt1465054361377
rm: `/user/cloudera/w261-output-3-2-1': No such file or directory
rm: cannot remove `/home/cloudera/w261/Outputs/Out_3_2-1': No such file
 or directory
16/06/04 08:32:57 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/06/04 08:32:57 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob7077780303199644l4.jar tmpDir=null
16/06/04 08:32:59 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 08:32:59 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 08:33:02 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 08:33:02 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 08:33:03 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0017
16/06/04 08:33:05 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0017
16/06/04 08:33:05 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0017/
16/06/04 08:33:06 INFO mapreduce.Job: Running job: job_1464634906532_00
17
16/06/04 08:33:24 INFO mapreduce.Job: Job job_1464634906532_0017 runnin
g in uber mode : false
16/06/04 08:33:24 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 08:34:27 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 08:34:54 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 08:34:57 INFO mapreduce.Job: Job job_1464634906532_0017 comple
ted successfully
16/06/04 08:34:57 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1448
                FILE: Number of bytes written=486673
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=61947
                HDFS: Number of bytes written=1093
                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=2
                Total time spent by all maps in occupied slots (ms)=153
65760
                Total time spent by all reduces in occupied slots (ms)=
6151808
                Total time spent by all map tasks (ms)=120045
                Total time spent by all reduce tasks (ms)=48061
                Total vcore-seconds taken by all map tasks=120045
                Total vcore-seconds taken by all reduce tasks=48061
                Total megabyte-seconds taken by all map tasks=15365760
                Total megabyte-seconds taken by all reduce tasks=615180
8
        Map-Reduce Framework
                Map input records=249
                Map output records=1083
                Map output bytes=11212
                Map output materialized bytes=1847
                Input split bytes=240
                Combine input records=1083
                Combine output records=159
                Reduce input groups=159
                Reduce shuffle bytes=1847
                Reduce input records=159
                Reduce output records=102
                Spilled Records=318
                Shuffled Maps =4
                Failed Shuffles=0
                Merged Map outputs=4
                GC time elapsed (ms)=1311
                CPU time spent (ms)=5560
                Physical memory (bytes) snapshot=501768192
                Virtual memory (bytes) snapshot=2943021056
                Total committed heap usage (bytes)=190840832
        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=61707
        File Output Format Counters
                Bytes Written=1093
16/06/04 08:34:57 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-1
```

Run the full data through Hadoop

In [288]:

```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/Consumer_Complaints.cs
v /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-2-1
!rm -r /home/cloudera/w261/Outputs/Out_3_2-1

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_2-1

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.map.tasks=2 -D mapred.reduce.tasks=2 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-combiner reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-2-1

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-2-1/* /home/clouder
a/w261/Outputs/Out_3_2-1
```

```
16/06/04 10:30:51 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/Consumer_Complaints.csv' to trash a
t: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cl
oudera/w261/Consumer_Complaints.csv1465061451380
16/06/04 10:31:00 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-2-1' to trash at: hdfs://qu
ickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-
output-3-2-1
16/06/04 10:31:03 INFO Configuration.deprecation: mapred.map.tasks is d
eprecated. Instead, use mapreduce.job.maps
16/06/04 10:31:03 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob4751754695751656259.jar tmpDir=null
16/06/04 10:31:05 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:31:06 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:31:07 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 10:31:07 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 10:31:07 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0022
16/06/04 10:31:08 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0022
16/06/04 10:31:08 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0022/
16/06/04 10:31:08 INFO mapreduce.Job: Running job: job_1464634906532_00
22
16/06/04 10:31:18 INFO mapreduce.Job: Job job_1464634906532_0022 runnin
g in uber mode : false
16/06/04 10:31:18 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 10:31:47 INFO mapreduce.Job:  map 25% reduce 0%
16/06/04 10:31:48 INFO mapreduce.Job:  map 59% reduce 0%
16/06/04 10:31:49 INFO mapreduce.Job:  map 75% reduce 0%
16/06/04 10:31:50 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 10:32:05 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 10:32:06 INFO mapreduce.Job: Job job_1464634906532_0022 comple
ted successfully
16/06/04 10:32:07 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=12127
                FILE: Number of bytes written=499029
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=50946999
                HDFS: Number of bytes written=2477
                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=2
                Total time spent by all maps in occupied slots (ms)=768
6400
```

```
		Total time spent by all reduces in occupied slots (ms)=
3129856
		Total time spent by all map tasks (ms)=60050
		Total time spent by all reduce tasks (ms)=24452
		Total vcore-seconds taken by all map tasks=60050
		Total vcore-seconds taken by all reduce tasks=24452
		Total megabyte-seconds taken by all map tasks=7686400
		Total megabyte-seconds taken by all reduce tasks=312985
6
	Map-Reduce Framework
		Map input records=312913
		Map output records=978634
		Map output bytes=10407527
		Map output materialized bytes=5984
		Input split bytes=252
		Combine input records=978634
		Combine output records=508
		Reduce input groups=508
		Reduce shuffle bytes=5984
		Reduce input records=508
		Reduce output records=182
		Spilled Records=1369
		Shuffled Maps =4
		Failed Shuffles=0
		Merged Map outputs=4
		GC time elapsed (ms)=1239
		CPU time spent (ms)=9540
		Physical memory (bytes) snapshot=473772032
		Virtual memory (bytes) snapshot=2924826624
		Total committed heap usage (bytes)=187170816
	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=50946747
	File Output Format Counters
		Bytes Written=2477
16/06/04 10:32:07 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-2-1
```

Pull out the top 50

In [289]:

```
# pull the two files that we outputted from
# hadoop
file1 = "Outputs/Out_3_2-1/part-00000"
file2 = "Outputs/Out_3_2-1/part-00001"

# set the lists for holding the words and
# the counts
words1 = []
counts1 = []
words2 = []
counts2 = []

# get the words from file 1
with open(file1, "r") as myfile:
    for line in myfile.readlines():

        # split the line
        line = line.split("\t")

        # set the word and the count
        word = line[0].strip()
        count = int(line[1].strip())

        # append to the lists
        words1.append(word)
        counts1.append(count)

# get the words from file 2
with open(file2, "r") as myfile:
    for line in myfile.readlines():

        # split the line
        line = line.split("\t")

        # set the word and the count
        word = line[0].strip()
        count = int(line[1].strip())

        # append to the lists
        words2.append(word)
        counts2.append(count)

# create a final list to store the counts
# and words
words = []
counts = []
LENGTH = 50

# while we have less than 50 items in
# our list
while len(words) < LENGTH:

    # compare the counts and words at
    # the end of each list
    word1 = words1[-1]
    count1 = counts1[-1]
    word2 = words2[-1]
    count2 = counts2[-1]

    # if word1 is larger, append it to the
    # final list
    if count1 > count2:
        words.append(words1.pop())
        counts.append(counts1.pop())

    # else if word is actually bigger, append
    # it to the final list
    elif count2 > count1:
        words.append(words2.pop())
        counts.append(counts2.pop())

    # else if the two words have equal counts
    else:

        # append both words to a new list and
        # sort that list
        sort = []
        sort.append(word1)
        sort.append(word2)
        sort.sort()

        # check which word is alphabetically
        # first and add that word to our
        # final list
        if word1 == sort[0]:
            words.append(words1.pop())
            counts.append(counts1.pop())
        else:
            words.append(words2.pop())
            counts.append(counts2.pop())

# print out the list of the top 50 words
for index,word in enumerate(words):
    info = word + "\t" + str(counts[index])
    print info
```

```
"Loan   107254
modification    70487
servicing       36767
credit  36126
report  30546
Incorrect       29069
information     29069
on      29069
or      22533
debt    17966
and     16448
"Account        16205
opening 16205
Credit  14768
club    12545
health  12545
/       12386
not     12353
loan    12237
collect 11848
attempts        11848
owed    11848
Cont'd  11848
of      10885
my      10731
Deposits        10555
withdrawals     10555
Problems        9484
"Application    8625
to      8401
Billing 8158
Other   7886
disputes        6938
Communication   6920
tactics 6920
reporting       6559
lease   6337
the     6248
by      5663
caused  5663
being   5663
funds   5663
low     5663
process 5505
verification    5214
Disclosure      5214
Managing        5006
company's       4858
investigation   4858
card    4405
```

HW 3.3 Shopping Cart Analysis

Product Recommendations: The action or practice of selling additional products or services to existing customers is called cross-selling. Giving product recommendation is one of the examples of cross-selling that are frequently used by online retailers. One simple method to give product recommendations is to recommend products that are frequently browsed together by the customers. For this homework use the online browsing behavior dataset located at:
https://www.dropbox.com/s/zlfyiwa70poqg74/ProductPurchaseData.txt?dl=0 (https://www.dropbox.com/s/zlfyiwa70poqg74/ProductPurchaseData.txt?dl=0)

Each line in this dataset represents a browsing session of a customer. On each line, each string of 8 characters represents the id of an item browsed during that session. The items are separated by spaces.
Here are the first few lines of the ProductPurchaseData:
FRO11987 ELE17451 ELE89019 SNA90258 GRO99222
GRO99222 GRO12298 FRO12685 ELE91550 SNA11465 ELE26917 ELE52966 FRO90334 SNA30755 ELE17451 FRO84225 SNA80192
ELE17451 GRO73461 DAI22896 SNA99873 FRO86643
ELE17451 ELE37798 FRO86643 GRO56989 ELE23393 SNA11465
ELE17451 SNA69641 FRO86643 FRO78087 SNA11465 GRO39357 ELE28573 ELE11375 DAI54444

Do some exploratory data analysis of this dataset guided by the following questions:
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.

Preprocess the data

We want to preprocess the file to add a unique identifier for each basket. This allows the products in each basket to be sorted but still retain their association with the original basket. This is a step similar to the one described when dealing with Microsoft.com's log files in the Async.

In [290]:
```
def Addkey(filename):
    """Adds a unique key to every row in a
    data file"""

    # open the original file
    with open(filename,"r") as myfile:

        # create a new filename
        newfilename = filename + "_mod"

        # create a new file for the found
        with open(newfilename,"w") as mynewfile:

            # initalize a record counter to assign each
            # basket it's own unique identifier
            count = 0

            # loop through all the lines
            for line in myfile.readlines():

                # create what we will be writing
                info = str(count) + "..\t.." + line

                # write to the new file
                mynewfile.write(info)

                # increment the count
                count = count + 1
```

In [291]:
```
# add a key to both my test and full data
Addkey("data/ProductPurchaseData_test.txt")
Addkey("data/ProductPurchaseData.txt")
print "Keys successfully added"
```

```
Keys successfully added
```

Mapper function

The mapper function takes the browsing session from a website and outputs each product visited and the number of products visited.

In [293]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.3

# import the system library to read from
# the input
import sys

# loop through each line
for line in sys.stdin:

    # split by the divider
    line = line.split("..\t..")

    # set the basket number and the items
    # in each basket
    basket_id = int(line[0])
    products = line[1].strip().split()

    # count the number of products in
    # the basket
    number = len(products)

    # print out the key-value pair as a tab
    # delimited list of word and
    for product in products:
        info = str(basket_id) + "\t" + \
        product.strip() + "," + str(number)
        print info
```

```
Overwriting mapper.py
```

Reducer function

The reducer function takes the output from the mapper and outputs:

- a list of products sorted by frequency of order, along with relative frequency
- number of unique products
- contents of the largest basket

In [294]:

```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.3

# import the system library to read from
# the input
# import the operator library to help sort
# the dictionary
import sys
import operator

# create a dictionary to store the products
# and their respective counts
products = {}

# create an array to hold the products in the
# largest basket, also add a marker to help
# us keep track of the items we add to the basket
basket = []
max_basket_id = None

# keep a running count of all products
all_prods = 0

# loop through each line
for line in sys.stdin:

    # split the line by the tab and set the
    # basket id
    line = line.split("\t")
    _basket_id = line[0]

    # set the product the next item
    # and the number of items as the one
    # after that
    _product = line[1].split(",")[0]
    _basket_total = int(line[1].split(",")[1])

    # check to see if the product already
    # exists in the dictionary
    if _product not in products:

        # if it's not, add it, and initalize
        # it with a count of 0
        products[_product] = 0

    # increment the counts
    products[_product] = products[_product] + 1
    all_prods = all_prods + 1

    # compare the length of the current basket
    # with the length of the basket currently stored
    if _basket_total > len(basket):

        # set how many items we want to add to
        # this basket
        max_basket_id = _basket_id

    # if we need to add items to the basket
    if _basket_id == max_basket_id:

        # add the item to the basket
        basket.append(_product)

# print the sorted list of products
for product in sorted(products.items(),\
                      key=lambda k: (-k[1], k[0])):
    freq = float(product[1])/float(all_prods)
    print product[0],"\t",product[1],"\t",freq

# print a dividing line
print "*~*~*~*~*"

# print the lenght of the largest basket
print "The number of items in the largest basket \
is", len(basket)

# print each item in the basket
info = ""
for _item in basket:
    info = info + str(_item) + " "
print info

# print a dividing line
print "*~*~*~*~*"

# print the number of unique products
print "Unique products:", len(products.keys())
```

```
Overwriting reducer.py
```

Make the files executable and run the test data hadoop

In [62]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/ProductPurchaseData_te
st.txt_mod /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-3
!rm -r /home/cloudera/w261/Outputs/Out_3_3

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_3

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-3

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-3/part-00000 /home/c
loudera/w261/Outputs/Out_3_3
```

```
16/05/30 15:43:48 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/ProductPurchase_mod.txt' to trash a
t: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cl
oudera/w261/ProductPurchase_mod.txt
16/05/30 15:43:56 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-3' to trash at: hdfs://quic
kstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-ou
tput-3-3
16/05/30 15:43:59 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob6949182719773429115.jar tmpDir=null
16/05/30 15:44:00 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 15:44:01 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/05/30 15:44:02 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/05/30 15:44:03 INFO mapreduce.JobSubmitter: number of splits:2
16/05/30 15:44:03 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0007
16/05/30 15:44:04 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0007
16/05/30 15:44:04 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0007/
16/05/30 15:44:04 INFO mapreduce.Job: Running job: job_1464634906532_00
07
16/05/30 15:44:27 INFO mapreduce.Job: Job job_1464634906532_0007 runnin
g in uber mode : false
16/05/30 15:44:27 INFO mapreduce.Job:  map 0% reduce 0%
16/05/30 15:45:21 INFO mapreduce.Job:  map 100% reduce 0%
16/05/30 15:45:31 INFO mapreduce.Job:  map 100% reduce 100%
16/05/30 15:45:31 INFO mapreduce.Job: Job job_1464634906532_0007 comple
ted successfully
16/05/30 15:45:31 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=9505
                FILE: Number of bytes written=380822
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=18582
                HDFS: Number of bytes written=13913
                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)=117
46944
                Total time spent by all reduces in occupied slots (ms)=
956160
                Total time spent by all map tasks (ms)=91773
                Total time spent by all reduce tasks (ms)=7470
                Total vcore-seconds taken by all map tasks=91773
                Total vcore-seconds taken by all reduce tasks=7470
                Total megabyte-seconds taken by all map tasks=11746944
                Total megabyte-seconds taken by all reduce tasks=956160
        Map-Reduce Framework
                Map input records=105
                Map output records=1265
                Map output bytes=18709
                Map output materialized bytes=9749
                Input split bytes=252
                Combine input records=0
                Combine output records=0
                Reduce input groups=105
                Reduce shuffle bytes=9749
                Reduce input records=1265
                Reduce output records=447
                Spilled Records=2530
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=3090
                CPU time spent (ms)=3960
                Physical memory (bytes) snapshot=361803776
                Virtual memory (bytes) snapshot=2188435456
                Total committed heap usage (bytes)=142606336
        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=18330
        File Output Format Counters
                Bytes Written=13913
16/05/30 15:45:31 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-3
```

Run the full data through Hadoop

In [295]:
```
# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/ProductPurchaseData.tx
t_mod /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-3
!rm -r /home/cloudera/w261/Outputs/Out_3_3

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_3

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-3

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-3/part-00000 /home/c
loudera/w261/Outputs/Out_3_3
```

```
16/06/04 10:45:35 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/ProductPurchaseData.txt_mod' to tras
h at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/use
r/cloudera/w261/ProductPurchaseData.txt_mod
16/06/04 10:45:44 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-3' to trash at: hdfs://quic
kstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-ou
tput-3-31465062344300
16/06/04 10:45:47 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob8907208976098100491.jar tmpDir=null
16/06/04 10:45:48 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:45:49 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/127.0.0.1:8032
16/06/04 10:45:50 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 10:45:50 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 10:45:51 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1464634906532_0024
16/06/04 10:45:51 INFO impl.YarnClientImpl: Submitted application appli
cation_1464634906532_0024
16/06/04 10:45:51 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1464634906532_0024/
16/06/04 10:45:51 INFO mapreduce.Job: Running job: job_1464634906532_00
24
16/06/04 10:46:03 INFO mapreduce.Job: Job job_1464634906532_0024 runnin
g in uber mode : false
16/06/04 10:46:03 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 10:46:34 INFO mapreduce.Job:  map 67% reduce 0%
16/06/04 10:46:36 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 10:46:50 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 10:46:51 INFO mapreduce.Job: Job job_1464634906532_0024 comple
ted successfully
16/06/04 10:46:51 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=3065209
                FILE: Number of bytes written=6467348
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3780013
                HDFS: Number of bytes written=394555
                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)=795
7248
                Total time spent by all reduces in occupied slots (ms)=
1406592
                Total time spent by all map tasks (ms)=62166
                Total time spent by all reduce tasks (ms)=10989
                Total vcore-seconds taken by all map tasks=62166
                Total vcore-seconds taken by all reduce tasks=10989
                Total megabyte-seconds taken by all map tasks=7957248
                Total megabyte-seconds taken by all reduce tasks=140659
2
        Map-Reduce Framework
                Map input records=31101
                Map output records=380824
                Map output bytes=6631793
                Map output materialized bytes=3040571
                Input split bytes=260
                Combine input records=0
                Combine output records=0
                Reduce input groups=31101
                Reduce shuffle bytes=3040571
                Reduce input records=380824
                Reduce output records=12597
                Spilled Records=761648
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=735
                CPU time spent (ms)=8560
                Physical memory (bytes) snapshot=376811520
                Virtual memory (bytes) snapshot=2195718144
                Total committed heap usage (bytes)=143654912
        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=3779753
        File Output Format Counters
                Bytes Written=394555
16/06/04 10:46:51 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-3
```

Handle the output data

The output data has the answer to multiple questions. We split the file into multiple files, one for each part of the question.

In [296]:
```
# set the divider information
divider = "*~*~*~*~*"
divider_count = 0

# initalize variables to hold the information
# for each future file
products = []
largest = []
uniques = []

# open the output file
with open("Outputs/Out_3_3/part-00000","r") \
as myfile:

    # loop through every line
    for line in myfile.readlines():

        # check to see if we've reached a divider
        if line.strip() == divider:
            divider_count = divider_count + 1
        else:

            # add the line to the appropriate file
            # based on the divider count
            if divider_count == 0:
                products.append(line)
            elif divider_count == 1:
                largest.append(line)
            else:
                uniques.append(line)

# write to each of the new files
with open("Outputs/Out_3_3/products","w") as myfile:
    for product in products:
        myfile.write(product)
with open("Outputs/Out_3_3/largest","w") as myfile:
    for line in largest:
        myfile.write(line)
with open("Outputs/Out_3_3/unqiues","w") as myfile:
    for line in uniques:
        myfile.write(line)
```

Number of unique products

```
In [297]: !cat Outputs/Out_3_3/unqiues
```

```
Unique products: 12592
```

Largest basket

```
In [298]: !cat Outputs/Out_3_3/largest
```

```
The number of items in the largest basket is 71
ELE89019 FRO11987 ELE17451 GRO99222 SNA90258 FRO12685 GRO12298 GRO99222
 SNA80192 FRO84225 ELE17451 SNA30755 FRO90334 ELE52966 ELE26917 SNA1146
5 ELE91550 DAI92253 DAI93692 SNA55952 GRO12935 GRO48282 DAI87514 SNA947
81 SNA17715 ELE82555 GRO36567 SNA47306 GRO99222 DAI22896 GRO73461 ELE17
451 DAI22177 ELE26917 FRO82427 ELE24180 GRO32086 ELE36890 ELE56095 SNA5
5762 SNA80324 SNA72462 ELE87243 DAI67621 ELE20847 SNA61380 FRO15030 DAI
75645 GRO99863 DAI38969 DAI62779 ELE56788 ELE81346 GRO94758 ELE49801 GR
O68067 SNA47306 ELE59028 GRO69543 DAI53152 FRO84460 GRO81087 GRO61133 D
AI85309 DAI84511 DAI54320 FRO37721 GRO46627 SNA96364 ELE35632 DAI67347
```

Top 50 products with their frequencies and relative frequencies

In [299]:
```
print "Product\t\tCount\tRelative Frequency"
!head -50 Outputs/Out_3_3/products
```

```
Product         Count   Relative Frequency
DAI62779        6667    0.0175067747831
FRO40251        3881    0.010191059387
ELE17451        3875    0.0101753040775
GRO73461        3602    0.00945843749344
SNA80324        3044    0.00799319370628
ELE32164        2851    0.0074863979161
DAI75645        2736    0.00718442114993
SNA45677        2455    0.0064465474865
FRO31317        2330    0.0061183118711
DAI85309        2293    0.00602115412894
ELE26917        2292    0.00601852824402
FRO80039        2233    0.00586360103355
GRO21487        2115    0.00555374661261
SNA99873        2083    0.00546971829507
GRO59710        2004    0.00526227338613
GRO71621        1920    0.00504169905258
FRO85978        1918    0.00503644728273
GRO30386        1840    0.00483162825872
ELE74009        1816    0.00476860702057
GRO56726        1784    0.00468457870302
DAI63921        1773    0.00465569396887
GRO46854        1756    0.00461105392517
ELE66600        1713    0.00449814087347
DAI83733        1712    0.00449551498855
FRO32293        1702    0.00446925613932
ELE66810        1697    0.0044561267147
SNA55762        1646    0.00432220658362
DAI22177        1627    0.00427231477008
FRO78087        1531    0.00402022981745
ELE99737        1516    0.0039808415436
ELE34057        1489    0.00390994265067
GRO94758        1489    0.00390994265067
FRO35904        1436    0.00377077074974
FRO53271        1420    0.00372875659097
SNA93860        1407    0.00369462008697
SNA90094        1390    0.00364998004327
GRO38814        1352    0.00355019641619
ELE56788        1345    0.00353181522173
GRO61133        1321    0.00346879398357
DAI88807        1316    0.00345566455896
ELE74482        1316    0.00345566455896
ELE59935        1311    0.00344253513434
SNA96271        1295    0.00340052097557
DAI43223        1290    0.00338739155095
ELE91337        1289    0.00338476566603
GRO15017        1275    0.0033480032771
DAI31081        1261    0.00331124088818
GRO81087        1220    0.00320357960633
DAI22896        1219    0.0032009537214
GRO85051        1214    0.00318782429679
```

HW3.4. (Computationally prohibitive but then again Hadoop can handle this) 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.

Mapper function

This mapper function takes the items in a basket and outputs pairs for each co-occurence for every term. For example, the line a,b,c would output: ab 1, ac 1, ba 1, bc 1, ca 1, cb 1. This is the pairs method.

In [3]:
```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.4

# import the system library to read from
# the input
import sys

# initalize counter to keep track of the number
# of baskets
baskets = 0

# write to standard error to keep track of how often
# we call this function
sys.stderr.write("reporter:counter:MyJob,Mapper,1\n")

# loop through each line
for line in sys.stdin:

    # increment the counter
    baskets = baskets + 1

    # split by spaces
    products = line.split()

    # loop through each product
    for product in products:

        # loop through every other product,
        # ignoring the the product itself
        # because we wouldn't want
        # to make a product recomendation for the
        # same product we're already visiting
        for pair in products:

            # if this is another product
            if product.strip() != pair.strip():

                # create the string to print
                info = product.strip() + "," + \
                pair.strip() + "\t1"
                print info

# print the counter as the last line
info = "Records:" + "\t" + str(baskets)
print info
```

```
Overwriting mapper.py
```

Reducer function

This function takes the output from the mappers and outputs in a sorted list for each co-occurring pair:

- item1
- item2
- support count (number of times this pair occurs)
- support (portion of baskets this pair occurs in)

In [5]:

```
%%writefile reducer.py
#!/home/cloudera/anaconda2/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.4

# import the system library to read from
# the input
# import the numpy library to help us sort
# before output
import sys
import numpy as np

# create a dictionary to store the product pairs
# and their respective counts
products = {}

# keep a running count of how many records we are
# dealing with
records = 0

# write a line to the standard error to keep
# track of how often this function is called
sys.stderr.write("reporter:counter:MyJob,Reducer,1\n")

# loop through each line
for line in sys.stdin:

    # split the line by the tab
    line = line.split("\t")

    # set the pair and count values
    pair = line[0].strip()
    count = int(line[1])

    # if its the count variable let's update
    # our counts
    if pair == "Records:":
        records = records + count

    else:
        # check to see if the pair already
        # exists in the dictionary
        if pair not in products:

            # if it's not, add it, and initalize
            # it with a count of 0
            products[pair] = 0

        # increment the counts
        products[pair] = products[pair] + 1

# create a set of array to store those pairs that
# frequently occur together
FREQUENT = 100
item1s = []
item2s = []
```

```
counts = []
frequs = []

# loop through all the pairs in the dictionary
for pair in products.keys():

    # if the pair occurs more than 100 times
    if int(products[pair]) > FREQUENT:

        # split the items and store them
        item1 = pair.split(',')[0]
        item2 = pair.split(',')[1]

        # get the count and relative
        # frequency
        count = products[pair]
        frequ = float(count)/float(records)

        # append all the values to our arrays
        item1s.append(item1)
        item2s.append(item2)
        counts.append(count)
        frequs.append(frequ)

# convert our arrays to numpy arrays
item1s = np.array(item1s)
item2s = np.array(item2s)
counts = np.array(counts)
frequs = np.array(frequs)

# get the indices for the sorted list
indexs = np.argsort(frequs)[::-1]

# print the elements to the output
print "Item1\t\tItem2\t\tCount\tSupport"
for index in indexs:

    # gather all the information
    item1 = item1s[index]
    item2 = item2s[index]
    count = counts[index]
    frequ = frequs[index]

    # create the line to print out
    line = str(item1) + "\t" + str(item2) + \
    "\t" + str(count) + "\t" + str(frequ)

    # print the line out
    print line
```

```
Overwriting reducer.py
```

Make the files executable and run the test data in hadoop

In [1]:
```
# write the time to a file
!echo $(date) > Outputs/Time_3_4

# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/ProductPurchaseData_te
st.txt /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-4
!rm -r /home/cloudera/w261/Outputs/Out_3_4

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_4

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-4

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-4/part-00000 /home/c
loudera/w261/Outputs/Out_3_4

# write the ending time to a file
!echo $(date) >> Outputs/Time_3_4
```

```
16/06/04 12:46:54 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/ProductPurchaseData_test.txt' to tra
sh at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/use
r/cloudera/w261/ProductPurchaseData_test.txt1465069614349
16/06/04 12:47:02 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-4' to trash at: hdfs://quic
kstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-ou
tput-3-41465069622054
16/06/04 12:47:04 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob2656552094061797701.jar tmpDir=null
16/06/04 12:47:05 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 12:47:05 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 12:47:06 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 12:47:06 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 12:47:07 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1465069396819_0001
16/06/04 12:47:07 INFO impl.YarnClientImpl: Submitted application appli
cation_1465069396819_0001
16/06/04 12:47:08 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1465069396819_0001/
16/06/04 12:47:08 INFO mapreduce.Job: Running job: job_1465069396819_00
01
16/06/04 12:47:17 INFO mapreduce.Job: Job job_1465069396819_0001 runnin
g in uber mode : false
16/06/04 12:47:17 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 12:47:29 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 12:47:36 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 12:47:37 INFO mapreduce.Job: Job job_1465069396819_0001 comple
ted successfully
16/06/04 12:47:37 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=36092
                FILE: Number of bytes written=434757
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=8046
                HDFS: Number of bytes written=28
                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)=249
0496
                Total time spent by all reduces in occupied slots (ms)=
654848
                Total time spent by all map tasks (ms)=19457
                Total time spent by all reduce tasks (ms)=5116
                Total vcore-seconds taken by all map tasks=19457
                Total vcore-seconds taken by all reduce tasks=5116
                Total megabyte-seconds taken by all map tasks=2490496
                Total megabyte-seconds taken by all reduce tasks=654848
        Map-Reduce Framework
                Map input records=50
                Map output records=7026
                Map output bytes=140504
                Map output materialized bytes=37094
                Input split bytes=262
                Combine input records=0
                Combine output records=0
                Reduce input groups=5207
                Reduce shuffle bytes=37094
                Reduce input records=7026
                Reduce output records=1
                Spilled Records=14052
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=180
                CPU time spent (ms)=3690
                Physical memory (bytes) snapshot=428154880
                Virtual memory (bytes) snapshot=2210664448
                Total committed heap usage (bytes)=142082048
        MyJob
                Mapper=2
                Reducer=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=7784
        File Output Format Counters
                Bytes Written=28
16/06/04 12:47:37 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-4
```

Run the whole data through Hadoop

In [6]:

```
# write the time to a file
!echo $(date) > Outputs/Time_3_4

# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/ProductPurchaseData.tx
t /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-4
!rm -r /home/cloudera/w261/Outputs/Out_3_4

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_4

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-4

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-4/part-00000 /home/c
loudera/w261/Outputs/Out_3_4

# write the ending time to a file
!echo $(date) >> Outputs/Time_3_4
```

```
16/06/04 12:54:34 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/ProductPurchaseData_test.txt' to tra
sh at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/use
r/cloudera/w261/ProductPurchaseData_test.txt1465070074023
16/06/04 12:54:40 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-4' to trash at: hdfs://quic
kstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-ou
tput-3-41465070080269
16/06/04 12:54:42 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob9078575781275816040.jar tmpDir=null
16/06/04 12:54:43 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 12:54:43 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 12:54:45 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 12:54:45 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 12:54:45 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1465069396819_0002
16/06/04 12:54:45 INFO impl.YarnClientImpl: Submitted application appli
cation_1465069396819_0002
16/06/04 12:54:45 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1465069396819_0002/
16/06/04 12:54:45 INFO mapreduce.Job: Running job: job_1465069396819_00
02
16/06/04 12:54:53 INFO mapreduce.Job: Job job_1465069396819_0002 runnin
g in uber mode : false
16/06/04 12:54:53 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 12:55:07 INFO mapreduce.Job:  map 51% reduce 0%
16/06/04 12:55:10 INFO mapreduce.Job:  map 59% reduce 0%
16/06/04 12:55:11 INFO mapreduce.Job:  map 77% reduce 0%
16/06/04 12:55:12 INFO mapreduce.Job:  map 83% reduce 0%
16/06/04 12:55:13 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 12:55:25 INFO mapreduce.Job:  map 100% reduce 83%
16/06/04 12:55:28 INFO mapreduce.Job:  map 100% reduce 93%
16/06/04 12:55:31 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 12:55:32 INFO mapreduce.Job: Job job_1465069396819_0002 comple
ted successfully
16/06/04 12:55:32 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=43244633
                FILE: Number of bytes written=62302003
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3498983
                HDFS: Number of bytes written=101764
                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)=442
```

```
8288
                Total time spent by all reduces in occupied slots (ms)=
2150144
                Total time spent by all map tasks (ms)=34596
                Total time spent by all reduce tasks (ms)=16798
                Total vcore-seconds taken by all map tasks=34596
                Total vcore-seconds taken by all reduce tasks=16798
                Total megabyte-seconds taken by all map tasks=4428288
                Total megabyte-seconds taken by all reduce tasks=215014
4
        Map-Reduce Framework
                Map input records=31101
                Map output records=5068110
                Map output bytes=101362190
                Map output materialized bytes=18550851
                Input split bytes=252
                Combine input records=0
                Combine output records=0
                Reduce input groups=1754191
                Reduce shuffle bytes=18550851
                Reduce input records=5068110
                Reduce output records=2623
                Spilled Records=15204330
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=200
                CPU time spent (ms)=31270
                Physical memory (bytes) snapshot=431661056
                Virtual memory (bytes) snapshot=2203160576
                Total committed heap usage (bytes)=144179200
        MyJob
                Mapper=2
                Reducer=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=3498731
        File Output Format Counters
                Bytes Written=101764
16/06/04 12:55:32 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-4
```

Show the output

In [7]:
```
!head -50 Outputs/Out_3_4/part-00000
```

```
Item1            Item2            Count    Support
ELE17451         DAI62779         1592     0.0511880646925
DAI62779         ELE17451         1592     0.0511880646925
FRO40251         SNA80324         1412     0.0454004694383
SNA80324         FRO40251         1412     0.0454004694383
FRO40251         DAI75645         1254     0.0403202469374
DAI75645         FRO40251         1254     0.0403202469374
FRO40251         GRO85051         1213     0.0390019613517
GRO85051         FRO40251         1213     0.0390019613517
GRO73461         DAI62779         1139     0.0366226166361
DAI62779         GRO73461         1139     0.0366226166361
DAI75645         SNA80324         1130     0.0363332368734
SNA80324         DAI75645         1130     0.0363332368734
FRO40251         DAI62779         1070     0.0344040384554
DAI62779         FRO40251         1070     0.0344040384554
DAI62779         SNA80324         923      0.0296775023311
SNA80324         DAI62779         923      0.0296775023311
DAI85309         DAI62779         918      0.0295167357963
DAI62779         DAI85309         918      0.0295167357963
GRO59710         ELE32164         911      0.0292916626475
ELE32164         GRO59710         911      0.0292916626475
GRO73461         FRO40251         882      0.0283592167454
DAI75645         DAI62779         882      0.0283592167454
DAI62779         DAI75645         882      0.0283592167454
FRO40251         GRO73461         882      0.0283592167454
DAI62779         ELE92920         877      0.0281984502106
ELE92920         DAI62779         877      0.0281984502106
FRO92469         FRO40251         835      0.026848011318
FRO40251         FRO92469         835      0.026848011318
ELE32164         DAI62779         832      0.0267515513971
DAI62779         ELE32164         832      0.0267515513971
GRO73461         DAI75645         712      0.0228931545609
DAI75645         GRO73461         712      0.0228931545609
ELE32164         DAI43223         711      0.022861001254
DAI43223         ELE32164         711      0.022861001254
GRO30386         DAI62779         709      0.02279669464
DAI62779         GRO30386         709      0.02279669464
ELE17451         FRO40251         697      0.0224108549564
FRO40251         ELE17451         697      0.0224108549564
ELE99737         DAI85309         659      0.0211890292917
DAI85309         ELE99737         659      0.0211890292917
DAI62779         ELE26917         650      0.020899649529
ELE26917         DAI62779         650      0.020899649529
GRO21487         GRO73461         631      0.0202887366966
GRO73461         GRO21487         631      0.0202887366966
SNA45677         DAI62779         604      0.0194205974084
DAI62779         SNA45677         604      0.0194205974084
ELE17451         SNA80324         597      0.0191955242597
SNA80324         ELE17451         597      0.0191955242597
DAI62779         GRO71621         595      0.0191312176457
```

Calculate the time the program took to run

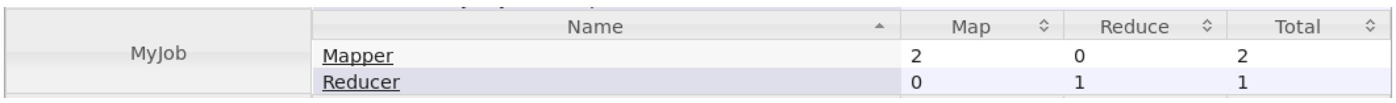
In [8]:
```
import datetime

def CalcTime(filepath):
    """A function that takes a simply file with
    only two lines, a starting and an ending time
    and returns the time elapsed between the two
    times"""

    # open the file
    with open(filepath,"r") as myfile:

        # set a starting and ending time
        start = None
        end = None

        # read in the lines and store them
        start = myfile.readline().strip()
        end = myfile.readline().strip()

        # set the time format
        time_format = "%a %b %d %H:%M:%S %Z %Y"

        # convert it to a time
        start = datetime.datetime.strptime(\
                                           start,\
                                           time_format)
        end = datetime.datetime.strptime(\
                                           end,\
                                           time_format)

        # calculate the elapsed time
        elapsed = end - start

        # return the elapsed time
        return elapsed
```

In [9]:
```
# calculate the elapsed time
elapsed_3_4 = CalcTime("Outputs/Time_3_4")
print "The time it took to run the program \
\nis",elapsed_3_4
```

```
The time it took to run the program
is 0:01:05
```

Counters for mappers and reducers

We used a total of 3 counters, 2 for mappers and 1 for reducers.



	Name	Map	Reduce	Total
MyJob	Mapper	2	0	2
	Reducer	0	1	1

HW 3.5

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.

Mapper function

The mapper function takes as input all items in a basket, each basket separated by a new line character. It outputs a dictionary key that holds the counts for each product and the different products it is paired with. For example, consider reading in the following list: a,b,c ; a,b ; c,d, ; a,b,e. Our output for product a will be: a (b:3, c:1, e:1).

In [1]:

```
%%writefile mapper.py
#!/usr/bin/python
## mapper.py
## Author: Alex Smith
## Description: mapper code for HW3.5

# import the system library to read from
# the input
import sys

# initalize counter to keep track of the number
# of baskets
baskets = 0

# initalize a dictionary to store each product
# and the counts for its paired products
prods = {}

# write to standard error to keep track of how often
# we call this function
sys.stderr.write("reporter:counter:MyJob,Mapper,1\n")

# loop through each line
for line in sys.stdin:

    # increment the counter
    baskets = baskets + 1

    # split by spaces
    products = line.split()

    # loop through each product
    for product in products:

        # strip away any excess
        product = product.strip()

        # if the product is not in already added
        # to the dictionary, add it
        if product not in prods.keys():
            prods[product]={}

        # loop through every other product,
        # ignoring the the product itself
        # because we wouldn't want
        # to make a product recomendation for the
        # same product we're already visiting
        for pair in products:

            # strip away any excess
            pair = pair.strip()

            # if this is another product
            if product != pair:

                # check to see if we already have
                # this pair in this product's
                # dictionary
                if pair not in prods[product].keys():
                    prods[product][pair] = 0

                # increment the count for the pairs
                # we've seen with this specific product
                prods[product][pair] = \
                prods[product][pair] + 1

# print the dictionary for each product
for product in prods.keys():
    info = product + "\t" + str(prods[product])
    print info

# print the counter as the last line
info = "Records:" + "\t" + str(baskets)
print info
```

```
Overwriting mapper.py
```

In [2]:
```
# test our mapper in the command line
!chmod +x mapper.py
!cat data/ProductPurchaseData_test.txt | ~/w261/mapper.py > Outputs/test
ing.txt
!head Outputs/testing.txt
```

```
reporter:counter:MyJob,Mapper,1
DAI50921        {'FRO78994': 1, 'GRO56989': 1, 'FRO32293': 1, 'FRO7121
3': 1, 'FRO18919': 1, 'SNA93730': 1, 'DAI36452': 1, 'ELE59935': 1, 'GRO
75578': 1, 'SNA14713': 1, 'GRO73461': 1, 'SNA58915': 1, 'GRO12935': 1,
 'DAI35347': 1, 'SNA81153': 1, 'SNA99873': 1, 'SNA45677': 1, 'SNA8019
2': 1, 'ELE17451': 2, 'SNA89670': 1, 'DAI22896': 1, 'FRO41069': 1, 'DAI
62779': 1}
FRO70974        {'GRO30386': 1, 'DAI88808': 1, 'GRO36567': 1, 'SNA9155
4': 1, 'SNA80192': 1, 'FRO16142': 1, 'ELE17451': 1, 'ELE52446': 1, 'ELE
96863': 1, 'DAI22896': 1, 'SNA47306': 1, 'SNA90258': 1, 'FRO18919': 1,
 'FRO41069': 1, 'GRO99222': 1, 'GRO49037': 1, 'FRO75418': 1, 'GRO7346
1': 1}
GRO94047        {'DAI22177': 1, 'FRO60023': 1, 'DAI59119': 1, 'FRO3131
7': 1, 'ELE89019': 1, 'ELE20196': 1, 'ELE17451': 1}
ELE13292        {'ELE12792': 1, 'DAI33885': 1, 'ELE27376': 1, 'DAI4435
5': 1, 'SNA56035': 1, 'ELE22574': 1, 'GRO71615': 1, 'ELE20166': 1, 'GRO
92942': 1}
ELE89019        {'GRO36567': 1, 'GRO56989': 1, 'GRO94047': 1, 'FRO3131
7': 1, 'ELE66067': 1, 'ELE14480': 1, 'FRO11987': 1, 'FRO36081': 1, 'DAI
93692': 1, 'ELE20196': 1, 'FRO60023': 1, 'SNA83730': 1, 'DAI22177': 1,
 'DAI59119': 1, 'SNA85034': 1, 'ELE73246': 1, 'ELE59935': 2, 'DAI2253
4': 1, 'SNA63157': 1, 'ELE37798': 1, 'FRO92261': 1, 'SNA90258': 1, 'FRO
75418': 1, 'ELE96863': 1, 'DAI91290': 1, 'DAI35347': 1, 'GRO39070': 1,
 'SNA55952': 1, 'DAI63921': 1, 'GRO82070': 1, 'SNA80192': 1, 'ELE1745
1': 5, 'FRO62970': 1, 'FRO79301': 1, 'DAI48891': 1, 'DAI62779': 1, 'GRO
99222': 2, 'GRO49037': 1, 'FRO78087': 2}
SNA11465        {'GRO56989': 2, 'GRO39369': 1, 'ELE11375': 1, 'DAI5444
4': 1, 'ELE91550': 1, 'FRO84225': 1, 'SNA69641': 1, 'FRO12685': 1, 'FRO
86643': 3, 'ELE23393': 1, 'FRO90334': 1, 'ELE52966': 1, 'ELE37798': 2,
 'GRO39357': 2, 'ELE28573': 1, 'SNA30755': 1, 'FRO78087': 2, 'GRO1229
8': 1, 'GRO75578': 1, 'ELE26917': 1, 'SNA80192': 1, 'ELE17451': 4, 'GRO
99222': 1, 'DAI95741': 1}
GRO27756        {'SNA99873': 1, 'ELE91337': 1, 'DAI69239': 1, 'ELE5296
6': 1, 'GRO99222': 1, 'SNA14713': 1, 'SNA93641': 1}
FRO36081        {'ELE96863': 1, 'FRO92261': 1, 'GRO56989': 1, 'FRO7541
8': 1, 'SNA80192': 1, 'ELE17451': 1, 'ELE89019': 1, 'ELE73246': 1, 'ELE
66067': 1, 'DAI48891': 1, 'DAI35347': 1, 'GRO36567': 1, 'SNA55952': 1,
 'DAI93692': 1, 'GRO49037': 1, 'SNA63157': 1, 'ELE37798': 1}
ELE20196        {'DAI22177': 1, 'DAI59119': 1, 'GRO94047': 1, 'FRO3131
7': 1, 'ELE89019': 1, 'FRO60023': 1, 'ELE17451': 1}
DAI84001        {'SNA45677': 1, 'GRO17442': 1, 'SNA77101': 1, 'SNA5891
5': 1, 'ELE91337': 1, 'DAI52318': 1, 'SNA30579': 1, 'ELE11468': 1, 'FRO
35353': 1, 'ELE20166': 1}
```

Reducer function

The reducer function takes the dictionaries outputted by the mapper and returns a sorted list for each co-occurring pair:

- item1
- item2
- support count (number of times this pair occurs)
- support (portion of baskets this pair occurs in)

In [7]:

```
%%writefile reducer.py
#!/usr/bin/python
## reducer.py
## Author: Alex Smith
## Description: reducer code for HW3.5

# import the system library to read from
# the input
# import the numpy library to help us sort
# before output
import sys
import numpy as np

# create a dictionary to store the product pairs
# and their respective counts
products = {}

# keep a running count of how many records we are
# dealing with
records = 0

# write a line to the standard error to keep
# track of how often this function is called
sys.stderr.write("reporter:counter:MyJob,Reducer,1\n")

# loop through each line
for line in sys.stdin:

    # split the line by the tab
    line = line.split("\t")

    # check to see if this is our records counter
    if line[0].strip() == "Records:":
        records = records + int(line[1])

    # otherwise we know its a pair we need to
    # process
    else:

        # set the first item
        item1 = line[0]

        # first let's convert it back into
        # a dictionary
        paired = eval(line[1])

        # loop through each of the paired words
        for pair in paired.keys():

            # set the count for the pair
            count = paired[pair]

            # create the pair
            pairs = item1 + "," + pair

            # check to see if it's already in
            # the dictionary
            if pairs not in products.keys():
                products[pairs] = 0

            # add to the count of the dictionary
            products[pairs] = products[pairs] + \
            count

# create a set of array to store those pairs that
# frequently occur together
FREQUENT = 100

# loop through all the pairs in the dictionary
for pair in products.keys():

    # if the pair occurs more than 100 times
    if int(products[pair]) > FREQUENT:

        # split the items and store them
        item1 = pair.split(',')[0]
        item2 = pair.split(',')[1]

        # get the count and relative
        # frequency
        count = products[pair]
        frequ = float(count)/float(records)

        # create the string to print
        info = item1 + "\t" + item2 + "\t" + \
        str(count) + "\t" + str(frequ)
        print info

# # convert our arrays to numpy arrays
# item1s = np.array(item1s)
# item2s = np.array(item2s)
# counts = np.array(counts)
# frequs = np.array(frequs)

# # get the indices for the sorted list
# indexs = np.argsort(frequs)[::-1]

# # print the elements to the output
# print "Item1\t\tItem2\t\tCount\tSupport"
# for index in indexs:

#     # gather all the information
#     item1 = item1s[index]
#     item2 = item2s[index]
#     count = counts[index]
#     frequ = frequs[index]

#     # create the line to print out
#     line = str(item1) + "\t" + str(item2) + \
#     "\t" + str(count) + "\t" + str(frequ)

#     # print the line out
#     print line
```

```
Overwriting reducer.py
```

In [54]:
```
# test the reducer with the mapper's output
# slightly modify the reducer to lower the
# frequency to 10
!chmod +x reducer.py
!cat Outputs/testing.txt | ~/w261/reducer.py
```

```
reporter:counter:MyJob,Reducer,1
ELE17451        GRO99222        15      0.3
ELE17451        GRO73461        11      0.22
GRO73461        ELE17451        11      0.22
GRO99222        ELE17451        15      0.3
```

Make the files executable and run the test data in Hadoop

In [ ]:
```
# write the current time to a file
!echo $(date) > Outputs/Time_3_5

# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/ProductPurchaseData_te
st2.txt /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-5
!rm -r /home/cloudera/w261/Outputs/Out_3_5

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_5

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-5

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-5/part-00000 /home/c
loudera/w261/Outputs/Out_3_5

# add the ending time to the file
!echo $(date) >> Outputs/Time_3_5
```

```
16/06/04 21:04:16 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/ProductPurchaseData.txt' to trash a
t: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/user/cl
oudera/w261/ProductPurchaseData.txt1465099456855
16/06/04 21:04:29 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-5' to trash at: hdfs://quic
kstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-ou
tput-3-51465099469400
16/06/04 21:04:34 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob5725157001918088713.jar tmpDir=null
16/06/04 21:04:35 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 21:04:36 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 21:04:38 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 21:04:38 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 21:04:39 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1465098020484_0003
16/06/04 21:04:39 INFO impl.YarnClientImpl: Submitted application appli
cation_1465098020484_0003
16/06/04 21:04:39 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1465098020484_0003/
16/06/04 21:04:39 INFO mapreduce.Job: Running job: job_1465098020484_00
03
16/06/04 21:04:59 INFO mapreduce.Job: Job job_1465098020484_0003 runnin
g in uber mode : false
16/06/04 21:04:59 INFO mapreduce.Job:  map 0% reduce 0%
```

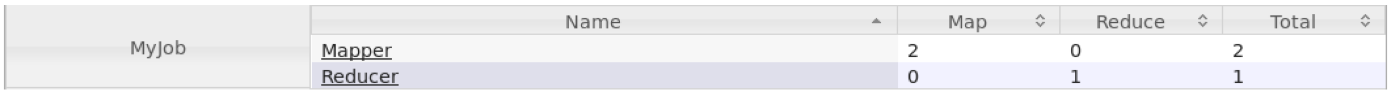
Run the full data through Hadoop

In [5]:
```
# write the current time to a file
!echo $(date) > Outputs/Time_3_5

# first let's clear our input directory to make
# sure that we're starting off with a clean slate
!hdfs dfs -rm -r /user/cloudera/w261/*

# modifies the permission to make the programs
# executable
!chmod +x ~/w261/mapper.py; chmod +x ~/w261/reducer.py

# put the input data into the hadoop cluster
!hdfs dfs -copyFromLocal /home/cloudera/w261/data/ProductPurchaseData_te
st.txt /user/cloudera/w261/

# make sure that we don't already have this output
!hdfs dfs -rm -r /user/cloudera/w261-output-3-5
!rm -r /home/cloudera/w261/Outputs/Out_3_5

# make the directory to store the output
!mkdir /home/cloudera/w261/Outputs/Out_3_5

# run the hadoop command
!hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stre
aming-mr1.jar \
-D mapred.reduce.tasks=1 \
-files /home/cloudera/w261/mapper.py,/home/cloudera/w261/reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-input /user/cloudera/w261/* -output /user/cloudera/w261-output-3-5

# copy the output files to the local directory
!hdfs dfs -copyToLocal /user/cloudera/w261-output-3-5/part-00000 /home/c
loudera/w261/Outputs/Out_3_5

# add the ending time to the file
!echo $(date) >> Outputs/Time_3_5
```

```
16/06/04 20:43:43 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261/ProductPurchaseData_test.txt' to tra
sh at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current/use
r/cloudera/w261/ProductPurchaseData_test.txt1465098223712
16/06/04 20:43:52 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstar
t.cloudera:8020/user/cloudera/w261-output-3-5' to trash at: hdfs://quic
kstart.cloudera:8020/user/cloudera/.Trash/Current/user/cloudera/w261-ou
tput-3-51465098232312
16/06/04 20:43:55 INFO Configuration.deprecation: mapred.reduce.tasks i
s deprecated. Instead, use mapreduce.job.reduces
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.7.0.jar] /tmp/
streamjob4761734599335409084.jar tmpDir=null
16/06/04 20:43:56 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 20:43:57 INFO client.RMProxy: Connecting to ResourceManager at
 quickstart.cloudera/10.0.2.15:8032
16/06/04 20:43:58 INFO mapred.FileInputFormat: Total input paths to pro
cess : 1
16/06/04 20:43:58 INFO mapreduce.JobSubmitter: number of splits:2
16/06/04 20:43:58 INFO mapreduce.JobSubmitter: Submitting tokens for jo
b: job_1465098020484_0001
16/06/04 20:43:59 INFO impl.YarnClientImpl: Submitted application appli
cation_1465098020484_0001
16/06/04 20:43:59 INFO mapreduce.Job: The url to track the job: http://
quickstart.cloudera:8088/proxy/application_1465098020484_0001/
16/06/04 20:43:59 INFO mapreduce.Job: Running job: job_1465098020484_00
01
16/06/04 20:44:09 INFO mapreduce.Job: Job job_1465098020484_0001 runnin
g in uber mode : false
16/06/04 20:44:09 INFO mapreduce.Job:  map 0% reduce 0%
16/06/04 20:44:22 INFO mapreduce.Job:  map 50% reduce 0%
16/06/04 20:44:23 INFO mapreduce.Job:  map 100% reduce 0%
16/06/04 20:44:33 INFO mapreduce.Job:  map 100% reduce 100%
16/06/04 20:44:34 INFO mapreduce.Job: Job job_1465098020484_0001 comple
ted successfully
16/06/04 20:44:34 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=28067
                FILE: Number of bytes written=415836
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=8046
                HDFS: Number of bytes written=102
                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)=294
5024
                Total time spent by all reduces in occupied slots (ms)=
963840
                Total time spent by all map tasks (ms)=23008
                Total time spent by all reduce tasks (ms)=7530
                Total vcore-seconds taken by all map tasks=23008
                Total vcore-seconds taken by all reduce tasks=7530
                Total megabyte-seconds taken by all map tasks=2945024
                Total megabyte-seconds taken by all reduce tasks=963840
        Map-Reduce Framework
                Map input records=50
                Map output records=268
                Map output bytes=86896
                Map output materialized bytes=26198
                Input split bytes=262
                Combine input records=0
                Combine output records=0
                Reduce input groups=219
                Reduce shuffle bytes=26198
                Reduce input records=268
                Reduce output records=4
                Spilled Records=536
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=212
                CPU time spent (ms)=2640
                Physical memory (bytes) snapshot=416141312
                Virtual memory (bytes) snapshot=2192523264
                Total committed heap usage (bytes)=143130624
        MyJob
                Mapper=2
                Reducer=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=7784
        File Output Format Counters
                Bytes Written=102
16/06/04 20:44:34 INFO streaming.StreamJob: Output directory: /user/clo
udera/w261-output-3-5
```

Use of counters

We used a total of 3 counters, 2 for the mappers and 1 for the reducer. We would have had more instances of the reducer counter had we used a combiner. However, in this case, we elected to not use a combiner.



	Name	Map	Reduce	Total
MyJob	Mapper	2	0	2
	Reducer	0	1	1

Calculate the elapsed time

In [231]:
```
elapsed_3_5 = CalcTime("Outputs/Time_3_5")
print "The elapsed time for the stripes approach \
\nis",elapsed_3_5
```

```
The elapsed time for the stripes approach
is 0:01:13
```

In [233]:
```
# let's compare the times
from prettytable import PrettyTable

# create a new table
pretty = PrettyTable(["Approach","Time"])
pretty.add_row(["Pairs",elapsed_3_4])
pretty.add_row(["Stripes",elapsed_3_5])

# show the table
print pretty
```

```
+----------+---------+
| Approach |   Time  |
+----------+---------+
|  Pairs   | 0:01:21 |
| Stripes  | 0:01:13 |
+----------+---------+
```