Colby Tobin
CS 6220

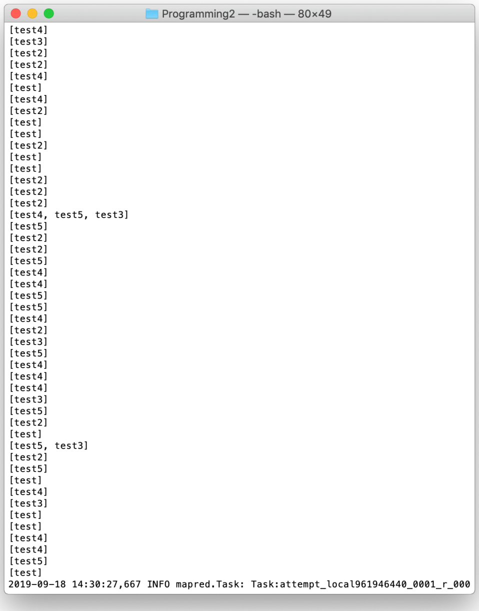
Assignment 2

Code Sources

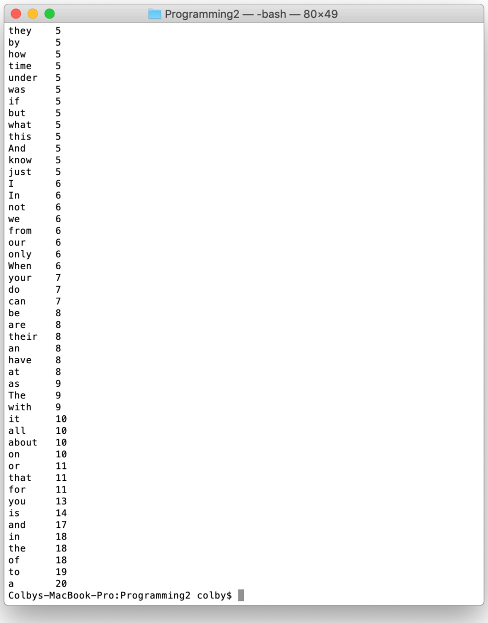
- Code and Hadoop Tutorial Found Here for the word count problem
- Basis of Code for Top 100 Words problem found Here
- GitHub link to executables and README for each of the problems Here

The rest of the project is on the next page and on the Excel document.

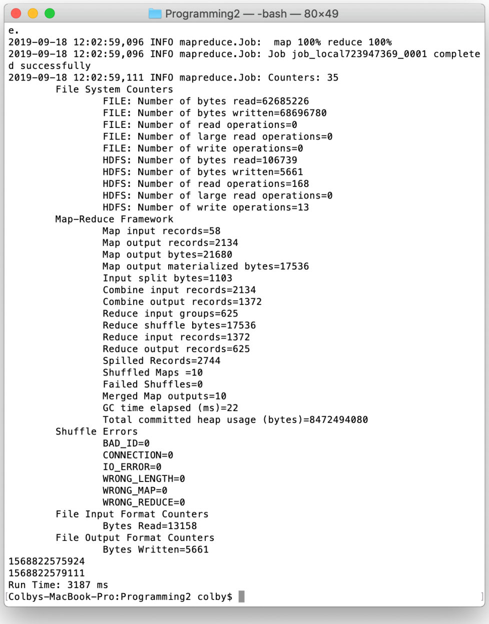
Screen Shot of Environment:



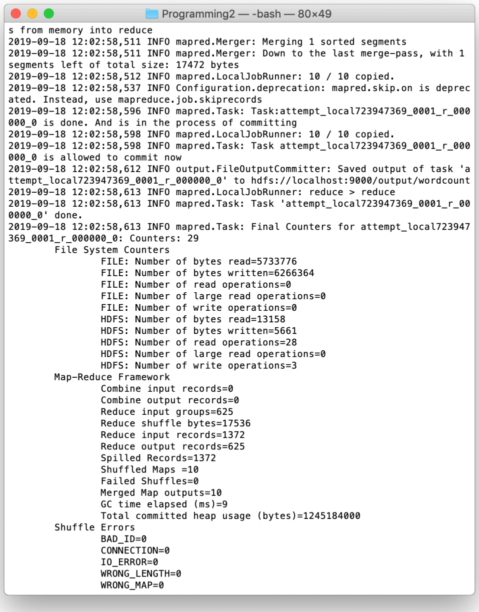
```
[test4]
[test3]
[test2]
[test2]
[test4]
[test]
[test4]
[test2]
[test]
[test]
[test2]
[test]
[test]
[test2]
[test2]
[test2]
[test4, test5, test3]
[test5]
[test2]
[test2]
[test5]
[test4]
[test4]
[test5]
[test5]
[test4]
[test2]
[test3]
[test5]
[test4]
[test4]
[test4]
[test3]
[test5]
[test2]
[test]
[test5, test3]
[test2]
[test5]
[test]
[test4]
[test3]
[test]
[test]
[test4]
[test4]
[test5]
[test]
2019-09-18 14:30:27,667 INFO mapred.Task: Task:attempt_local961946440_0001_r_000
```



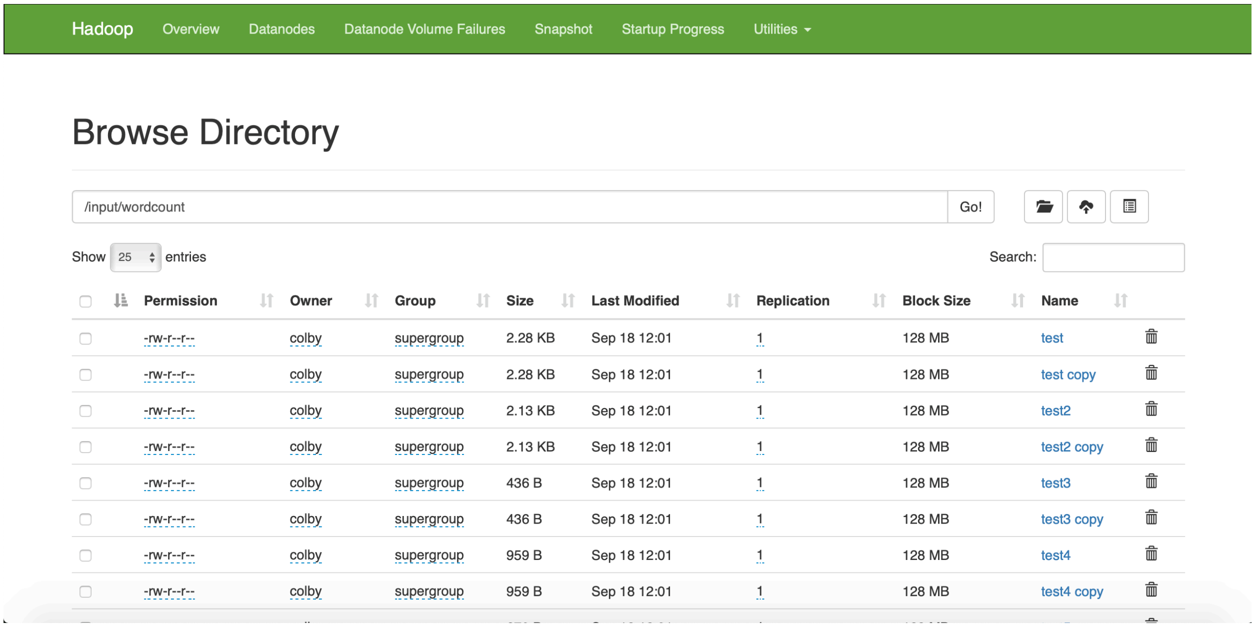
```
they    5
by      5
how     5
time    5
under   5
was     5
if      5
but     5
what    5
this    5
And     5
know    5
just    5
I       6
In      6
not     6
we      6
from    6
our     6
only    6
When    6
your    7
do      7
can     7
be      8
are     8
their   8
an      8
have    8
at      8
as      9
The     9
with    9
it      10
all     10
about   10
on      10
or      11
that    11
for     11
you     13
is      14
and     17
in      18
the     18
of      18
to      19
a       20
Colbys-MacBook-Pro:Programming2 colby$ 
```



```
e.
2019-09-18 12:02:59,096 INFO mapreduce.Job:  map 100% reduce 100%
2019-09-18 12:02:59,096 INFO mapreduce.Job: Job job_local723947369_0001 complete
d successfully
2019-09-18 12:02:59,111 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=62685226
                FILE: Number of bytes written=68696780
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=106739
                HDFS: Number of bytes written=5661
                HDFS: Number of read operations=168
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=13
        Map-Reduce Framework
                Map input records=58
                Map output records=2134
                Map output bytes=21680
                Map output materialized bytes=17536
                Input split bytes=1103
                Combine input records=2134
                Combine output records=1372
                Reduce input groups=625
                Reduce shuffle bytes=17536
                Reduce input records=1372
                Reduce output records=625
                Spilled Records=2744
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=22
                Total committed heap usage (bytes)=8472494080
        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=13158
        File Output Format Counters
                Bytes Written=5661
1568822575924
1568822579111
Run Time: 3187 ms
Colbys-MacBook-Pro:Programming2 colby$ 
```



```
s from memory into reduce
2019-09-18 12:02:58,511 INFO mapred.Merger: Merging 1 sorted segments
2019-09-18 12:02:58,511 INFO mapred.Merger: Down to the last merge-pass, with 1
segments left of total size: 17472 bytes
2019-09-18 12:02:58,512 INFO mapred.LocalJobRunner: 10 / 10 copied.
2019-09-18 12:02:58,537 INFO Configuration.deprecation: mapred.skip.on is deprec
ated. Instead, use mapreduce.job.skiprecords
2019-09-18 12:02:58,596 INFO mapred.Task: Task:attempt_local723947369_0001_r_000
000_0 is done. And is in the process of committing
2019-09-18 12:02:58,598 INFO mapred.LocalJobRunner: 10 / 10 copied.
2019-09-18 12:02:58,598 INFO mapred.Task: Task attempt_local723947369_0001_r_000
000_0 is allowed to commit now
2019-09-18 12:02:58,612 INFO output.FileOutputCommitter: Saved output of task 'a
ttempt_local723947369_0001_r_000000_0' to hdfs://localhost:9000/output/wordcount
2019-09-18 12:02:58,613 INFO mapred.LocalJobRunner: reduce > reduce
2019-09-18 12:02:58,613 INFO mapred.Task: Task 'attempt_local723947369_0001_r_00
0000_0' done.
2019-09-18 12:02:58,613 INFO mapred.Task: Final Counters for attempt_local723947
369_0001_r_000000_0: Counters: 29
        File System Counters
                FILE: Number of bytes read=5733776
                FILE: Number of bytes written=6266364
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=13158
                HDFS: Number of bytes written=5661
                HDFS: Number of read operations=28
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=3
        Map-Reduce Framework
                Combine input records=0
                Combine output records=0
                Reduce input groups=625
                Reduce shuffle bytes=17536
                Reduce input records=1372
                Reduce output records=625
                Spilled Records=1372
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=9
                Total committed heap usage (bytes)=1245184000
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
```



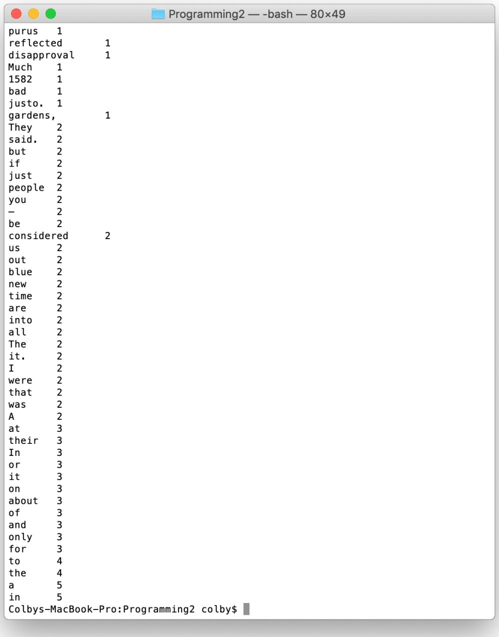
Browse Directory

/input/wordcount

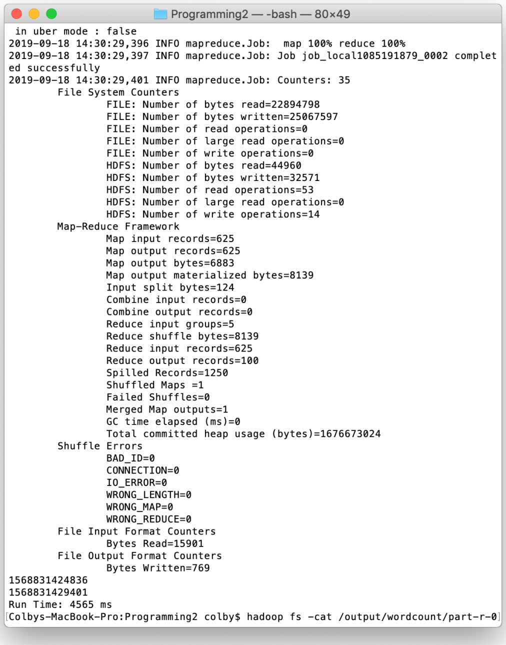
Show 25 entries

Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	colby	supergroup	2.28 KB	Sep 18 12:01	1	128 MB	test
□	-rw-r--r--	colby	supergroup	2.28 KB	Sep 18 12:01	1	128 MB	test copy
□	-rw-r--r--	colby	supergroup	2.13 KB	Sep 18 12:01	1	128 MB	test2
□	-rw-r--r--	colby	supergroup	2.13 KB	Sep 18 12:01	1	128 MB	test2 copy
□	-rw-r--r--	colby	supergroup	436 B	Sep 18 12:01	1	128 MB	test3
□	-rw-r--r--	colby	supergroup	436 B	Sep 18 12:01	1	128 MB	test3 copy
□	-rw-r--r--	colby	supergroup	959 B	Sep 18 12:01	1	128 MB	test4
□	-rw-r--r--	colby	supergroup	959 B	Sep 18 12:01	1	128 MB	test4 copy



```
purus   1
reflected       1
disapproval     1
Much    1
1582    1
bad     1
justo.  1
gardens,        1
They    2
said.   2
but     2
if      2
just    2
people  2
you     2
–       2
be      2
considered      2
us      2
out     2
blue    2
new     2
time    2
are     2
into    2
all     2
The     2
it.     2
I       2
were    2
that    2
was     2
A       2
at      3
their   3
In      3
or      3
it      3
on      3
about   3
of      3
and     3
only    3
for     3
to      4
the     4
a       5
in      5
Colbys-MacBook-Pro:Programming2 colby$ 
```



```
 in uber mode : false
2019-09-18 14:30:29,396 INFO mapreduce.Job:  map 100% reduce 100%
2019-09-18 14:30:29,397 INFO mapreduce.Job: Job job_local1085191879_0002 complet
ed successfully
2019-09-18 14:30:29,401 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=22894798
                FILE: Number of bytes written=25067597
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=44960
                HDFS: Number of bytes written=32571
                HDFS: Number of read operations=53
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=14
        Map-Reduce Framework
                Map input records=625
                Map output records=625
                Map output bytes=6883
                Map output materialized bytes=8139
                Input split bytes=124
                Combine input records=0
                Combine output records=0
                Reduce input groups=5
                Reduce shuffle bytes=8139
                Reduce input records=625
                Reduce output records=100
                Spilled Records=1250
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=0
                Total committed heap usage (bytes)=1676673024
        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=15901
        File Output Format Counters
                Bytes Written=769
1568831424836
1568831429401
Run Time: 4565 ms
Colbys-MacBook-Pro:Programming2 colby$ hadoop fs -cat /output/wordcount/part-r-0
```

Data for Runtime Statistics

The data for all of the runtime statistics is provided in the excel document that has been attached as a submission file. They are not in this document because the data would not fit in the space provided.

Problems Chosen:

1. The first problem is WordCount.

a. The variable that changed amongst the 2 experiments was the number of files that were being analyzed (10 files and 20 files).

b. Each of the 2 experiments were run 3 times and the data was recorded in the excel document.

c. Input of problem

  i. An input directory where the files that were to be analyzed were located

    1. These files contained passages gathered from the internet.

  ii. An output directory that pointed where the output file should be written

    1. This file contained the word counts for each word

d. Output of Problem

  i. It wrote to a file listed in the output directory. This file contained the word counts for each of the words in the input files

  ii. It also created a blank file that represented the success of the job

2. The second problem is Top 100 words appearing in 20 files. It was solved by 3 different programs

   a. The first program solved the problem using 1 job with 1 mapper and 1 reducer. The mapper mapped words to files, while the reducer found the top 100 words appearing in those files

   b. The second program solved the problem using 2 jobs with 2 mappers and 2 reducers. The first job mapped words to files, which was then reduced to a key value pair of the word to the array of files, and then stored in a file for use by the second job. The second job mapped the count of those files to a word, which was then reduced to the top 100 words that appeared in the files.

   c. The third program solved the problem using 2 jobs with 2 mappers and 2 reducers without utilizing any arrays for writing. The first job mapped words to files, which was then reduced to a key value pair of the word to number of files it appeared in, and then stored in a file for use by the second job. The second job mapped the count of those files to a word, which was then reduced to the top 100 words that appeared in the files.

   d. Input of Problem:

      i. An input directory where the files that were to be analyzed were located

         1. These files contained passages gathered from the internet.

      ii. An output directory that pointed where the output file should be written

         1. This file contained the top 100 words that existed within the input files

   e. Output of Problem

      i. 1 job program

         1. It wrote to a file listed in the output directory. This file contained the top 100 words that existed within the input files

         2. It also created a blank file that represented the success of the job

      ii. 2 jobs programs

         1. It wrote to a file listed in the output directory. This file contained the top 100 words that existed within the input files

         2. It also created a blank file that represented the success of the job for each of the 2 jobs

3. It wrote to a temporary directory listed as */input/wordcount/indicies* as defined in the program which held a mapping of the word to either a list of files or the number of files the word appeared in depending on which program it was.

Observations and Conclusions

Word Count Problem

The 2 different experiments run for the word count problem included a dataset with 10 files and a dataset with 20 files. Each of the files was relatively of equal size. When looking at the runtime of each of these experiments, it can be seen that the program with 20 files took a longer time to run the program. However, the dataset was scaled by a factor of 2. Based on this, one would assume that the runtime of the program would be scaled by a factor similar to 2, but in reality the program with 20 files resulted in a runtime that was only 1.3 times longer than the program with 10 files. With this in mind, it shows that this map-reduce program scales fairly effectively as the runtime is not necessarily proportional to the number of files that are being analyzed as input.

When looking at other statistics gathered from this experiment, most of them are fairly trivial for this problem. The program with 20 files took up more space than the program with 10 files and required more blocks of data to be allocated because of the increased file size, which is a given for this problem. The program with 20 files required 22 blocks whereas the program with 10 files only required 12 blocks of data for the files. Additionally, when looking at the total mapper time and total reducer time, it can be seen that the total mapper time for the program with 20 files is longer than the total mapper time for the program with 10 files due to more files that need to be analyzed. However, when comparing the average total reducer time, it can be seen that the average reducer times are very similar. This likely stems from the fact that the reducer is iterating over a fairly similar map of objects from the mapper, so it results in a fairly similar runtime performance.

Top 100 Words Problem

There were 3 different experiments run for the top 100 words problem. These included a program with 1 job, a program with 2 jobs that wrote arrays to a file, and a program with 2 jobs that wrote numbers to a file instead of arrays. When comparing the runtime performances of the different programs, it can be seen that the program with 1 job ran the fastest. This is likely due to the fact that the one job was only dependent on the input files. In the programs with 2 jobs, the first job had to finish with the mapper and reducer and output the data to a file before the second job could begin. Because of this, the jobs could not be performed asynchronously, so this wait time between the 2 jobs accounted for some of the delay in time. Additionally, because there were 2 jobs, 2 sets of files had to be read as input, which resulted in a longer runtime. When comparing the programs that each had 2 jobs, it makes sense that the program that did not write arrays to files worked faster. Serializing an array and writing it to a fiel takes a much longer time than just writing a serializable integer to a file. Additionally, reading an array takes a much longer time than reading an integer as well (*O(n) vs. O(1)).* With both of these reasons in mind, this demonstrates why the program that did not write an array to a file resulted in a faster performance than the other program with 2 jobs. Arrays are costly, so avoiding their use improves the performance of the program.

When looking at the other runtime statistics, one can see other interesting data points that arise. When comparing the total load and number of files required by the system, it can be seen that the 1 job program requires less memory. This is due to the fact that there is no interim file that needs to be created to store temporary data with 1 job. The 1 job only requires input files and a single output file. The 2 jobs require a set of input files, a temporary output file for the first job, and an overall output file for the second job. This results in requiring more data blocks as well as more files in the system for the programs with 2 jobs. When comparing the programs with 2 jobs, it can be seen that the total load is less for the program that does not utilize arrays. This is due to the fact that arrays require more data and space for allocation in a file, which is why those files hold more data and produce a higher load on the system. Lastly, when looking at the total mapper and reducer times, it can be seen that the program with 1 job clearly has the lowest mapper and reducer time due to their only being 1 mapper and 1 reducer that only has to read from files once and write to files once. With the programs with 2 jobs, they all have to read from files twice and write to files twice which increases their mapper and reducer times. When looking at the programs with 2 jobs, the total mapper times are very similar and there is not much distinction. However, when comparing the reducer times, it can be seen that the program that utilizes arrays has a much larger reducer time. This is likely because the reducer is in charge of writing to the output file, so in the case of the program with the arrays, the reducer must write the entire array to the output file, which takes more time than writing an integer, which is why there is a longer reducer time for that program.