Programming Assignment 2: MapReduce on Azure

By: Joe Dorris and Drew Masters

Part 1:

For part 1, we made a standard map-reduce word count in python that would write out the counts to a text file (word_count/mapper.py,word_count/reducer.py). We started a hadoop cluster on Azure and then loaded the complete works of shakespeare onto the cluster. We then ran the word count example and wrote the output to text file.

```
hadoop fs -copyFromLocal 100.txt.utf-8 ///example/100.txt.utf-8
yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar -files
mapper.py,reducer.py -mapper mapper.py -reducer reducer.py -input
wasb:///example/100.txt.utf-8 -output wasb:///example/word_counts
```

We sorted this file and examined the most frequent words (word_count/word_counts_sorted.txt). We determined that the first 40 words were stop words because they were articles or pronouns. So these were hardcoded in our inverted index map-reduce python scripts to be ignored. They are listed in word_count/stop_words.txt.

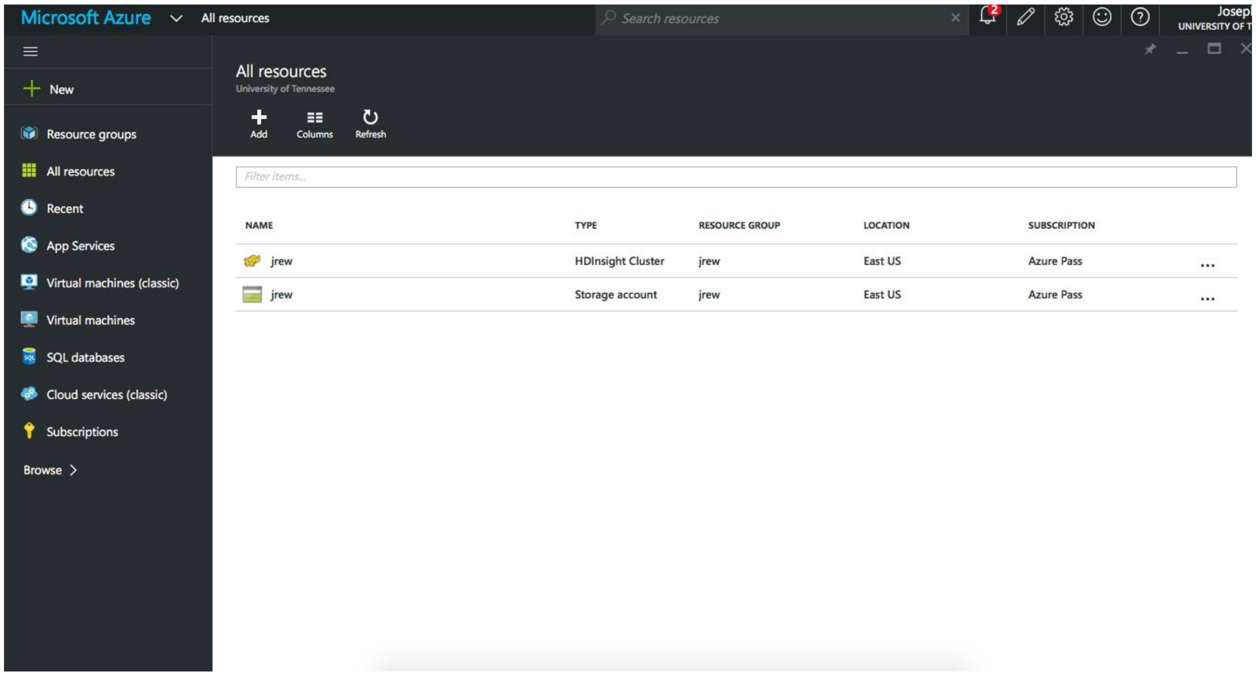
Part 2:

For this part, we determined that the files would need the document number and the line number appended to each line to be able to process it. We downloaded the complete works of Shakespeare and Romeo and Juliet and then appended each line in the file with the appropriate information (documents/shakespeare_ln.txt, documents/romeo_ln.txt). We then wrote a inverted_index/mapper_inverted_index.py which reads in the document number and line number and then writes out this information along with the position for each word in the line that is not a stop word. We then wrote inverted_index/reducer_inverted_index.py which takes all of the outputs, sorts them, and appends them to a list for each word. This is then written to a file for indexing.

To test, we copied the files to the hadoop file system and then ran hadoop streaming with our mapper and reducer.

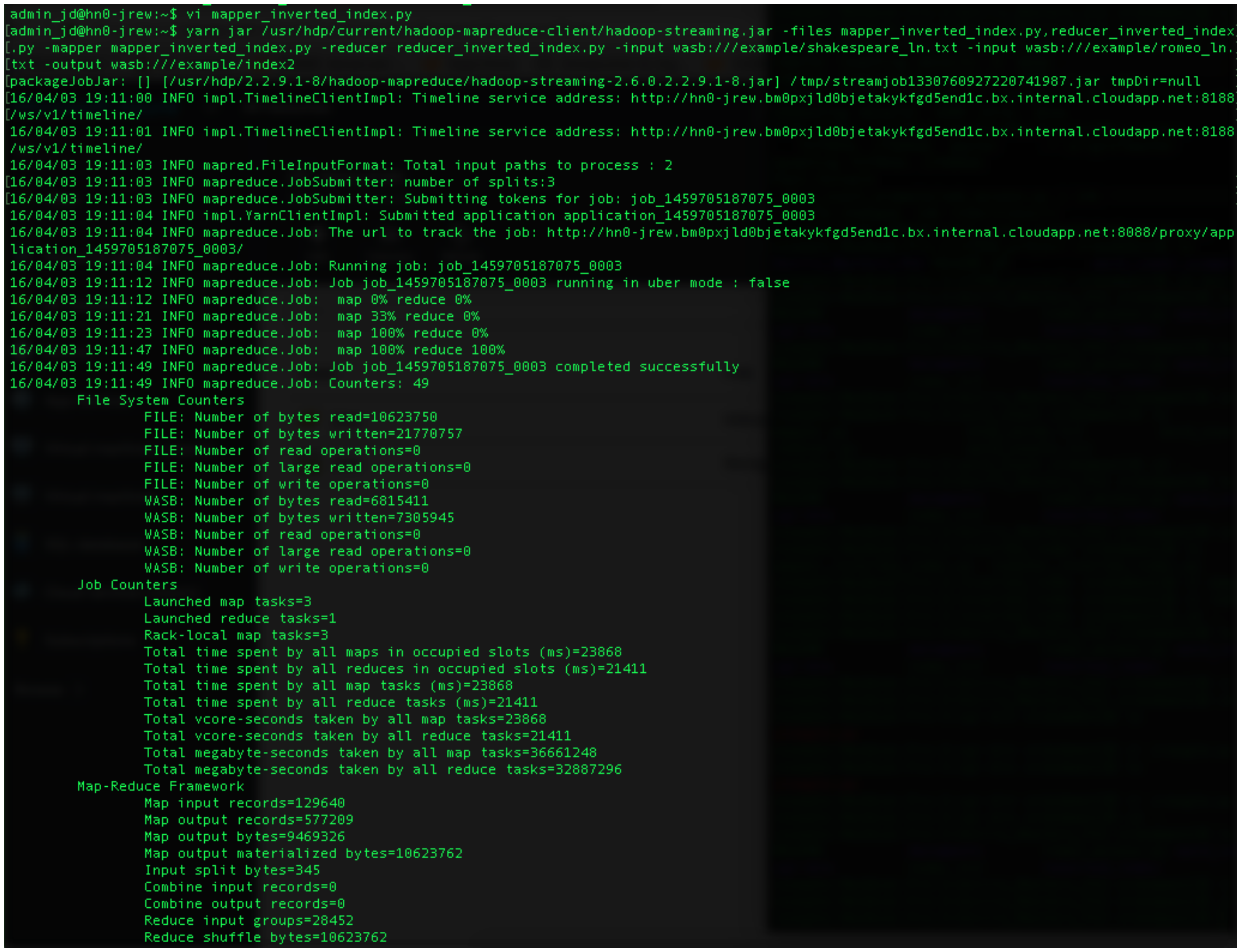
```
   16  hadoop fs -copyFromLocal shakespeare_ln.txt ///example/shakespeare_ln.txt
   17  hadoop fs -copyFromLocal romeo_ln.txt ///example/romeo_ln.txt
   18  yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar -files
mapper_inverted_index.py,reducer_inverted_index.py -mapper mapper_inverted_index.py
-reducer reducer_inverted_index.py -input wasb:///example/shakespeare_ln.txt -input
wasb:///example/romeo_ln.txt -output wasb:///example/index
```

The results are in index.txt.



All resources
University of Tennessee

NAME	TYPE	RESOURCE GROUP	LOCATION	SUBSCRIPTION
jrew	HDInsight Cluster	jrew	East US	Azure Pass
jrew	Storage account	jrew	East US	Azure Pass



```
 admin_jd@hn0-jrew:~$ vi mapper_inverted_index.py
[admin_jd@hn0-jrew:~$ yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar -files mapper_inverted_index.py,reducer_inverted_index
[.py -mapper mapper_inverted_index.py -reducer reducer_inverted_index.py -input wasb:///example/shakespeare_ln.txt -input wasb:///example/romeo_ln.
[txt -output wasb:///example/index2
[packageJobJar: [] [/usr/hdp/2.2.9.1-8/hadoop-mapreduce/hadoop-streaming-2.6.0.2.2.9.1-8.jar] /tmp/streamjob1330760927220741987.jar tmpDir=null
[16/04/03 19:11:00 INFO impl.TimelineClientImpl: Timeline service address: http://hn0-jrew.bm0pxjld0bjetakykfgd5end1c.bx.internal.cloudapp.net:8188
[/ws/v1/timeline/
 16/04/03 19:11:01 INFO impl.TimelineClientImpl: Timeline service address: http://hn0-jrew.bm0pxjld0bjetakykfgd5end1c.bx.internal.cloudapp.net:8188
 /ws/v1/timeline/
 16/04/03 19:11:03 INFO mapred.FileInputFormat: Total input paths to process : 2
[16/04/03 19:11:03 INFO mapreduce.JobSubmitter: number of splits:3
[16/04/03 19:11:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1459705187075_0003
 16/04/03 19:11:04 INFO impl.YarnClientImpl: Submitted application application_1459705187075_0003
 16/04/03 19:11:04 INFO mapreduce.Job: The url to track the job: http://hn0-jrew.bm0pxjld0bjetakykfgd5end1c.bx.internal.cloudapp.net:8088/proxy/app
 lication_1459705187075_0003/
 16/04/03 19:11:04 INFO mapreduce.Job: Running job: job_1459705187075_0003
 16/04/03 19:11:12 INFO mapreduce.Job: Job job_1459705187075_0003 running in uber mode : false
 16/04/03 19:11:12 INFO mapreduce.Job:  map 0% reduce 0%
 16/04/03 19:11:21 INFO mapreduce.Job:  map 33% reduce 0%
 16/04/03 19:11:23 INFO mapreduce.Job:  map 100% reduce 0%
 16/04/03 19:11:47 INFO mapreduce.Job:  map 100% reduce 100%
 16/04/03 19:11:49 INFO mapreduce.Job: Job job_1459705187075_0003 completed successfully
 16/04/03 19:11:49 INFO mapreduce.Job: Counters: 49
         File System Counters
                 FILE: Number of bytes read=10623750
                 FILE: Number of bytes written=21770757
                 FILE: Number of read operations=0
                 FILE: Number of large read operations=0
                 FILE: Number of write operations=0
                 WASB: Number of bytes read=6815411
                 WASB: Number of bytes written=7305945
                 WASB: Number of read operations=0
                 WASB: Number of large read operations=0
                 WASB: Number of write operations=0
         Job Counters
                 Launched map tasks=3
                 Launched reduce tasks=1
                 Rack-local map tasks=3
                 Total time spent by all maps in occupied slots (ms)=23868
                 Total time spent by all reduces in occupied slots (ms)=21411
                 Total time spent by all map tasks (ms)=23868
                 Total time spent by all reduce tasks (ms)=21411
                 Total vcore-seconds taken by all map tasks=23868
                 Total vcore-seconds taken by all reduce tasks=21411
                 Total megabyte-seconds taken by all map tasks=36661248
                 Total megabyte-seconds taken by all reduce tasks=32887296
         Map-Reduce Framework
                 Map input records=129640
                 Map output records=577209
                 Map output bytes=9469326
                 Map output materialized bytes=10623762
                 Input split bytes=345
                 Combine input records=0
                 Combine output records=0
                 Reduce input groups=28452
                 Reduce shuffle bytes=10623762
```

Part 3:
To use this file to search, we created a python script (index_access.py) that reads the data into a dictionary. A user is then prompted to enter a word to search and this dictionary is searched and the results are returned.

Extra Credit:
We wrote a python CGI script (cgi-bin/jrewgle.py) that acts as a web portal to the search. It consists of a form which takes the query and then searches the dictionary and returns the results. A server can be started using “python3 -m http.server 8080” in the main directory. It can then be reached using “http://localhost:8080/cgi-bin/jrewgle.py” in a browser. It does not support multiple words, “and”, “or”, or “not”.