Foster Mosden
Big Data Assignment 5

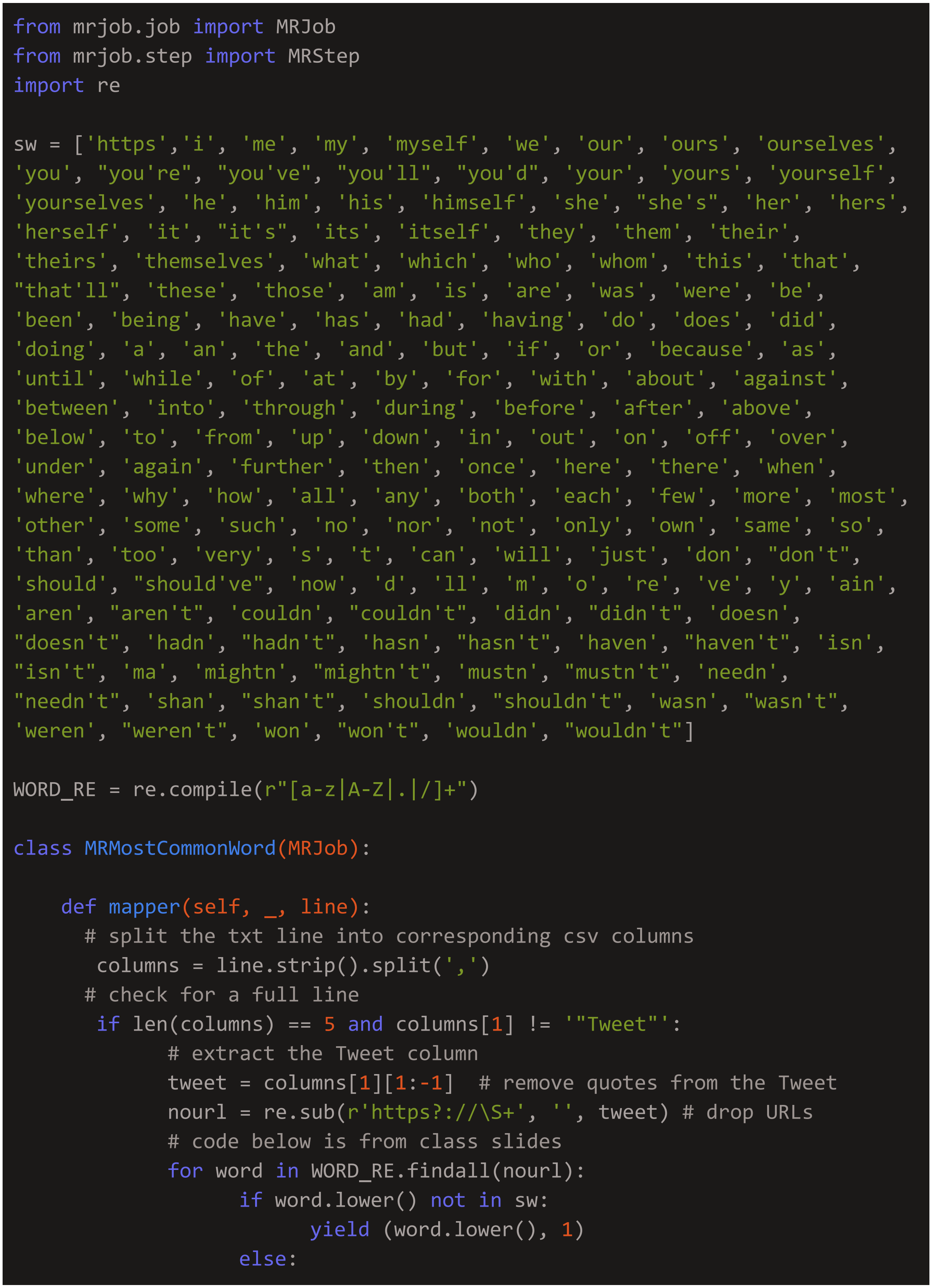
If a customer searches for tweets using the keyword “black friday,” MapReduce will be used to process the data and get search results from it.

First, MapReduce will split the input data into smaller chunks. Each split would be distributed across the Hadoop cluster. Each node is then tasked with processing the data that it is holding locally. The mapper tasks on these nodes are responsible for scanning through these input splits and identifying which tweets contain “black friday.” The output is a key-value pair where the key is the content and data of the tweet, with the value being an integer indicating the number of times “black friday” appeared in the tweet.

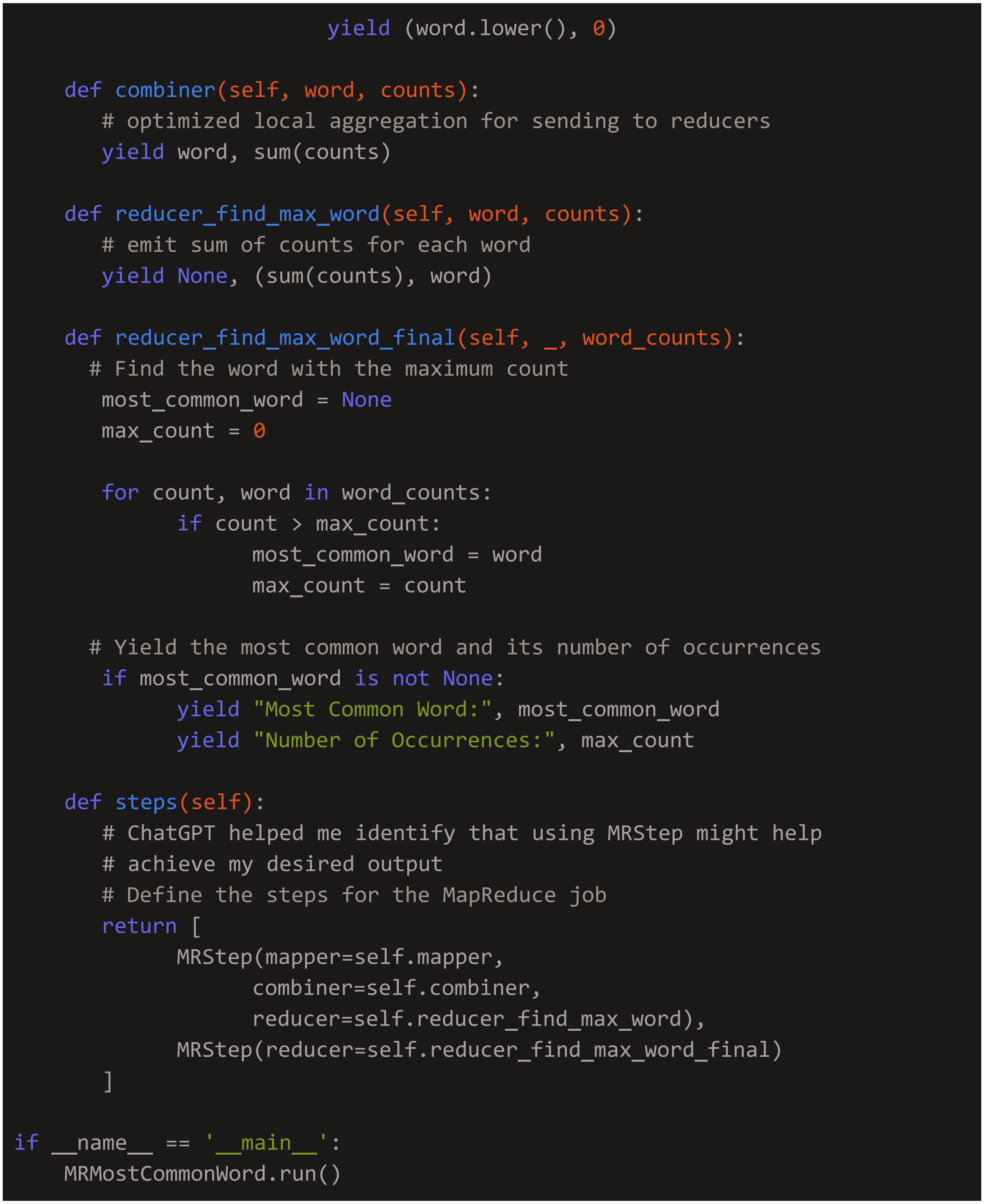
Then, the key-value pairs are shuffled and sorted by key (tweet). The main goal of this step is efficiency, as we want to group all values for each key together and ensure they are all sent to the same reducer in the next phase.

Next, is reducing. Our reducers will receive these sorted key-value pairs, count the number of tweets with the word “black friday”, and maintain a list of tweets along with the counts of the keyword associated. Each reducer will then select the top 10 tweets (based on whatever metrics specified -- remember these are stored along with the tweet text as the key in the key-value pair). Ultimately, each reducer outputs the total count of tweets with the keyword, and the top 10 tweets found by the reducer.

Finally, the search results are returned by the reducers, which aggregate this data and calculate a total tweet count, as well as selecting the ultimate top 10 tweets. Overall, this is an extremely efficient manner of searching, as splitting up the workload across various nodes and functions allows giant datasets (such as the entire history of all tweets) to be searched efficiently by many users.



```
from mrjob.job import MRJob
from mrjob.step import MRStep
import re

sw = ['https','i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves',
'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers',
'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that',
"that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be',
'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did',
'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as',
'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against',
'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when',
'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most',
'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't",
'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain',
'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn',
"doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn',
"isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn',
"needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't",
'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

WORD_RE = re.compile(r"[a-z|A-Z|.|/]+")

class MRMostCommonWord(MRJob):

    def mapper(self, _, line):
      # split the txt line into corresponding csv columns
       columns = line.strip().split(',')
      # check for a full line
       if len(columns) == 5 and columns[1] != '"Tweet"':
             # extract the Tweet column
             tweet = columns[1][1:-1]  # remove quotes from the Tweet
             nourl = re.sub(r'https?://\S+', '', tweet) # drop URLs
             # code below is from class slides
             for word in WORD_RE.findall(nourl):
                   if word.lower() not in sw:
                         yield (word.lower(), 1)
                   else:
```



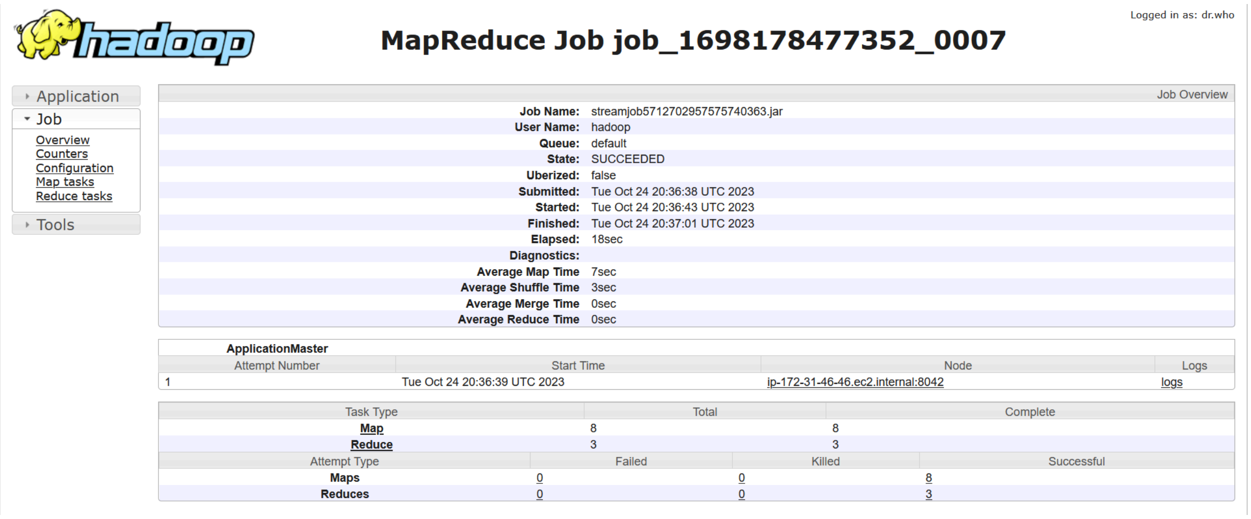
```
                         yield (word.lower(), 0)

    def combiner(self, word, counts):
       # optimized local aggregation for sending to reducers
       yield word, sum(counts)

    def reducer_find_max_word(self, word, counts):
       # emit sum of counts for each word
       yield None, (sum(counts), word)

    def reducer_find_max_word_final(self, _, word_counts):
      # Find the word with the maximum count
       most_common_word = None
       max_count = 0

       for count, word in word_counts:
             if count > max_count:
                   most_common_word = word
                   max_count = count

      # Yield the most common word and its number of occurrences
       if most_common_word is not None:
             yield "Most Common Word:", most_common_word
             yield "Number of Occurrences:", max_count

    def steps(self):
       # ChatGPT helped me identify that using MRStep might help
       # achieve my desired output
       # Define the steps for the MapReduce job
       return [
             MRStep(mapper=self.mapper,
                   combiner=self.combiner,
                   reducer=self.reducer_find_max_word),
             MRStep(reducer=self.reducer_find_max_word_final)
       ]

if __name__ == '__main__':
    MRMostCommonWord.run()
```

FOR ALL TWEETS:

Output:

"Most Common Word:"	*"rt"*
"Number of Occurrences:"	*382*

MapReduce completion screenshot:



MapReduce Job job_1698178477352_0007

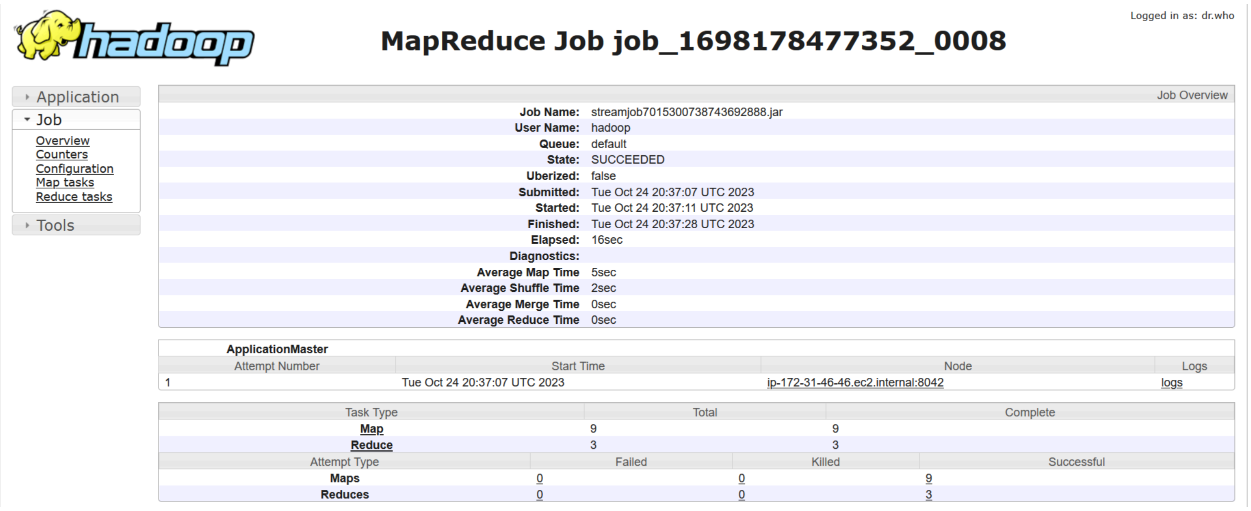
Job Name:	streamjob5712702957575740363.jar
User Name:	hadoop
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Tue Oct 24 20:36:38 UTC 2023
Started:	Tue Oct 24 20:36:43 UTC 2023
Finished:	Tue Oct 24 20:37:01 UTC 2023
Elapsed:	18sec
Diagnostics:	
Average Map Time	7sec
Average Shuffle Time	3sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Oct 24 20:36:39 UTC 2023	ip-172-31-46-46.ec2.internal:8042	logs

Task Type	Total	Complete
Map	8	8
Reduce	3	3

Attempt Type	Failed	Killed	Successful
Maps	0	0	8
Reduces	0	0	3



MapReduce Job job_1698178477352_0008

Job Name:	streamjob7015300738743692888.jar
User Name:	hadoop
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Tue Oct 24 20:37:07 UTC 2023
Started:	Tue Oct 24 20:37:11 UTC 2023
Finished:	Tue Oct 24 20:37:28 UTC 2023
Elapsed:	16sec
Diagnostics:	
Average Map Time	5sec
Average Shuffle Time	2sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Oct 24 20:37:07 UTC 2023	ip-172-31-46-46.ec2.internal:8042	logs

Task Type	Total	Complete
Map	9	9
Reduce	3	3

Attempt Type	Failed	Killed	Successful
Maps	0	0	9
Reduces	0	0	3

Links:

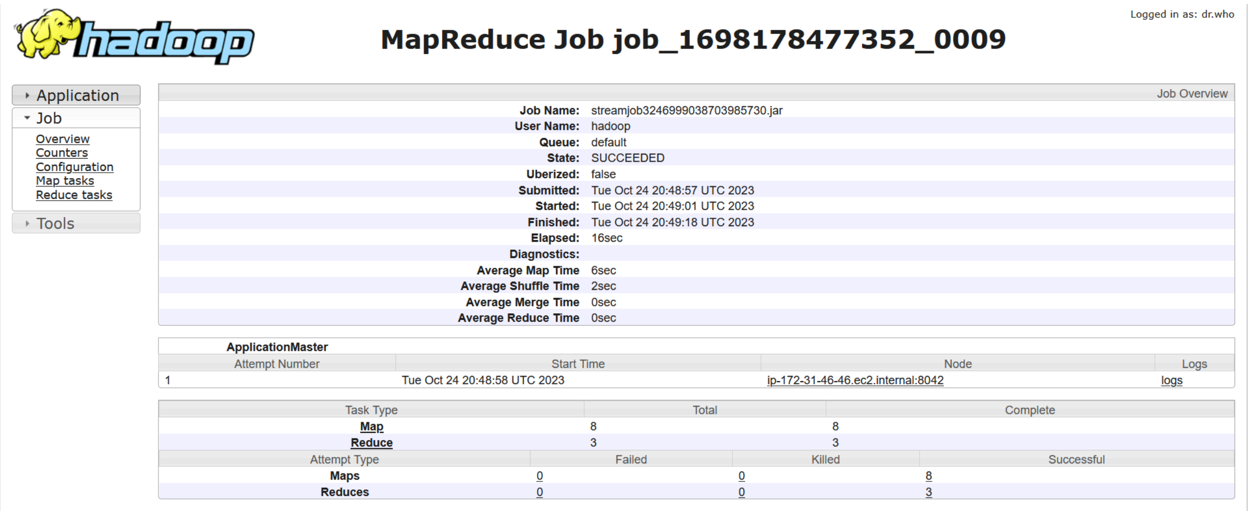
http://ip-172-31-40-203.ec2.internal:19888/jobhistory/job/job_1698178477352_0007
http://ip-172-31-40-203.ec2.internal:19888/jobhistory/job/job_1698178477352_0008

FOR FIRST 10 TWEETS:

Output:

"Most Common Word:" "anywhere"
"Number of Occurrences:" 2

MapReduce completion screenshot:



Logged in as: dr.who

MapReduce Job job_1698178477352_0009

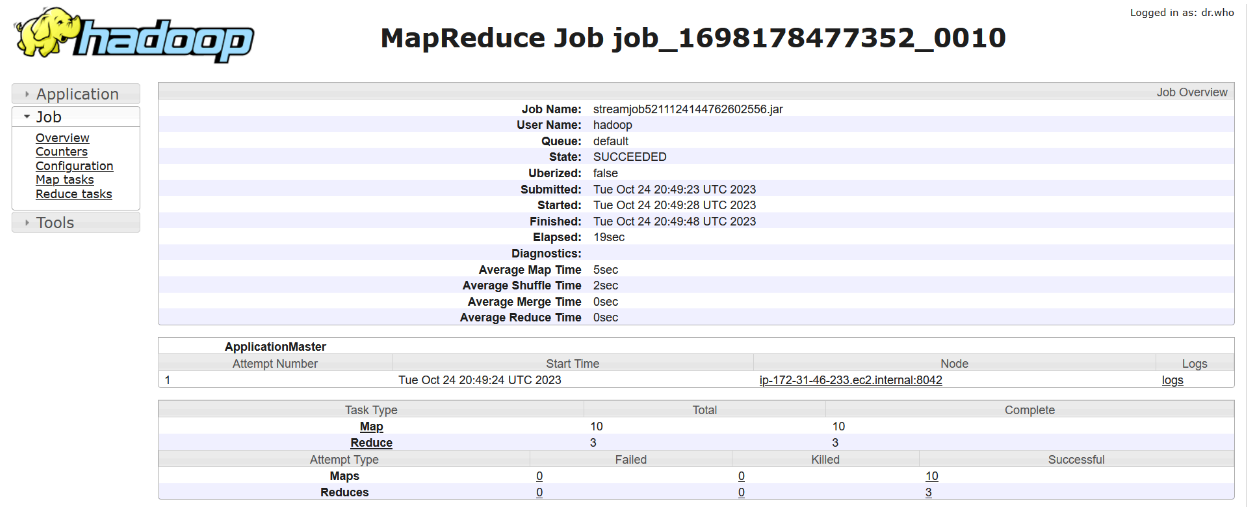
	Job Overview
Job Name:	streamjob3246999038703985730.jar
User Name:	hadoop
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Tue Oct 24 20:48:57 UTC 2023
Started:	Tue Oct 24 20:49:01 UTC 2023
Finished:	Tue Oct 24 20:49:18 UTC 2023
Elapsed:	16sec
Diagnostics:	
Average Map Time	6sec
Average Shuffle Time	2sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Oct 24 20:48:58 UTC 2023	ip-172-31-46-46.ec2.internal:8042	logs

Task Type	Total	Complete
Map	8	8
Reduce	3	3

Attempt Type	Failed	Killed	Successful
Maps	0	0	8
Reduces	0	0	3



Logged in as: dr.who

MapReduce Job job_1698178477352_0010

	Job Overview
Job Name:	streamjob5211124144762602556.jar
User Name:	hadoop
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Tue Oct 24 20:49:23 UTC 2023
Started:	Tue Oct 24 20:49:28 UTC 2023
Finished:	Tue Oct 24 20:49:48 UTC 2023
Elapsed:	19sec
Diagnostics:	
Average Map Time	5sec
Average Shuffle Time	2sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Oct 24 20:49:24 UTC 2023	ip-172-31-46-233.ec2.internal:8042	logs

Task Type	Total	Complete
Map	10	10
Reduce	3	3

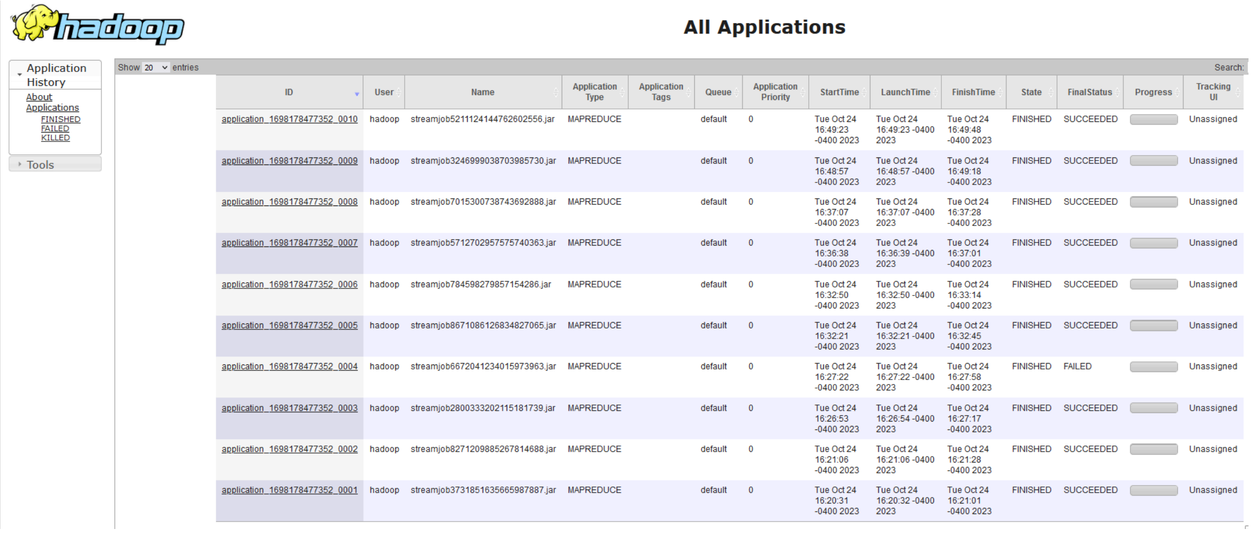
Attempt Type	Failed	Killed	Successful
Maps	0	0	10
Reduces	0	0	3

Links:

http://ip-172-31-40-203.ec2.internal:19888/jobhistory/job/job_1698178477352_0009
http://ip-172-31-40-203.ec2.internal:19888/jobhistory/job/job_1698178477352_0010

YARN Tasks View:

I had to run this a few times before I got the responses I needed - the last four rows are what I submitted for this assignment.



All Applications

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1698178477352_0010	hadoop	streamjob5211124144762602556.jar	MAPREDUCE		default	0	Tue Oct 24 16:49:23 -0400 2023	Tue Oct 24 16:49:23 -0400 2023	Tue Oct 24 16:49:48 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0009	hadoop	streamjob3246999038703985730.jar	MAPREDUCE		default	0	Tue Oct 24 16:48:57 -0400 2023	Tue Oct 24 16:48:57 -0400 2023	Tue Oct 24 16:49:18 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0008	hadoop	streamjob7015300738743692888.jar	MAPREDUCE		default	0	Tue Oct 24 16:37:07 -0400 2023	Tue Oct 24 16:37:07 -0400 2023	Tue Oct 24 16:37:28 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0007	hadoop	streamjob5712702957575740363.jar	MAPREDUCE		default	0	Tue Oct 24 16:36:38 -0400 2023	Tue Oct 24 16:36:39 -0400 2023	Tue Oct 24 16:37:01 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0006	hadoop	streamjob784598279857154286.jar	MAPREDUCE		default	0	Tue Oct 24 16:32:50 -0400 2023	Tue Oct 24 16:32:50 -0400 2023	Tue Oct 24 16:33:14 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0005	hadoop	streamjob8671086126834827065.jar	MAPREDUCE		default	0	Tue Oct 24 16:32:21 -0400 2023	Tue Oct 24 16:32:21 -0400 2023	Tue Oct 24 16:32:45 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0004	hadoop	streamjob6672041234015973963.jar	MAPREDUCE		default	0	Tue Oct 24 16:27:22 -0400 2023	Tue Oct 24 16:27:22 -0400 2023	Tue Oct 24 16:27:58 -0400 2023	FINISHED	FAILED		Unassigned
application_1698178477352_0003	hadoop	streamjob2800333202115181739.jar	MAPREDUCE		default	0	Tue Oct 24 16:26:53 -0400 2023	Tue Oct 24 16:26:54 -0400 2023	Tue Oct 24 16:27:17 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0002	hadoop	streamjob8271209885267814688.jar	MAPREDUCE		default	0	Tue Oct 24 16:21:06 -0400 2023	Tue Oct 24 16:21:06 -0400 2023	Tue Oct 24 16:21:28 -0400 2023	FINISHED	SUCCEEDED		Unassigned
application_1698178477352_0001	hadoop	streamjob3731851635665987887.jar	MAPREDUCE		default	0	Tue Oct 24 16:20:31 -0400 2023	Tue Oct 24 16:20:32 -0400 2023	Tue Oct 24 16:21:01 -0400 2023	FINISHED	SUCCEEDED		Unassigned

Running my code on all Tweets vs. the first 10:

For the dataset of all tweets, the first MR job took 8 mappers and 3 reducers in 18 seconds, while the second MR job took 9 mappers and 3 reducers in 16 seconds. While the number of mappers and reducers remained relatively stable, the runtime of the second job was slightly faster, likely due to the smaller dataset (which was the output of the first MR job). On the other hand, for the dataset of the first 10 tweets, the first MR job also used 8 mappers and 3 reducers, similar to the full dataset. The second MR job, however, used 10 mappers instead of 9, and 3 reducers, taking 19 seconds to complete, which is slightly longer than the first job. This could be due to a few reasons. For one, the increase in mappers might suggest that there was more data skew or other non-uniform distribution of words, leading to a more complex workload distribution among the mappers. It may also suggest that the use of additional mappers was ill conceived. When running a MapReduce job on HDFS, the number of mappers determines how many chunks the input data is divided into for parallel processing, and there is a delicate balance when it comes to employing mappers. At a certain point, adding more mappers ceases to lead to a decrease in runtime for each mapper, and thus each additional mapper only adds to the runtime, rather than shortening it. Overall, the number of mappers and reducers was relatively consistent between the two datasets, with runtime differences mainly influenced by the dataset size, and potentially also impacted by data skew/poor resource allocation.