1. Done individually.

2. Repository Links
   - MapReduce: CS6240/hw-1-mapreduce-jill666666: hw-1-mapreduce-jill666666 created by GitHub Classroom
   - Spark: CS6240/hw-1-spark-jill666666: hw-1-spark-jill666666 created by GitHub Classroom

3. Pseudo-code for the Twitter-follower-count program in MapReduce

```
TWITTER-FOLLOWER-COUNT

""" Map. """
def map(input_string):
  # given the input string, split by the whitespace "\n"
  # e.g., "1,2\n3,4" -> tokens = ["1,2","3,4"]
  tokens = input_string.split("\n")

  # iterate over the tokens to get the userID being followed
  for token in tokens:
    # e.g., "1,2" -> userID = "2"
    userID = token.split(",")[1];

    # emit the resulting pair (user ID, 1)
    emit (userID, 1)

""" Reduce. """
def reduce(key, value):
  # key = user ID, value = array of counts e.g., [1, 1, 1, ...]

  # iterate over the value to accumulate the count
  total_count = 0
  for count in value:
    total_count += count

  # if the total_count is divisible by 100 (remainder is 0),
  # write the result to the output
  if total_count % 100 == 0:
    output <- write (key, total_count)
```

4. Solution discussion

The map function first splits the input string by whitespace (“\n”) to store each line in the array ‘tokens’. We then iterate through these tokens to get the pair (userID, 1). Here, the ‘userID’ is equal to the ID being followed, since we split the token by “,” and selected the second element of the resulting pair (follower ID, ID being followed). The map function eventually emits the pair (userID, 1).

Reduce function takes in the key (userID) and value (counts). The program iterates through the value to get the total count. If the total count is divisible by 100, the pair (key, total count) is written to the output.

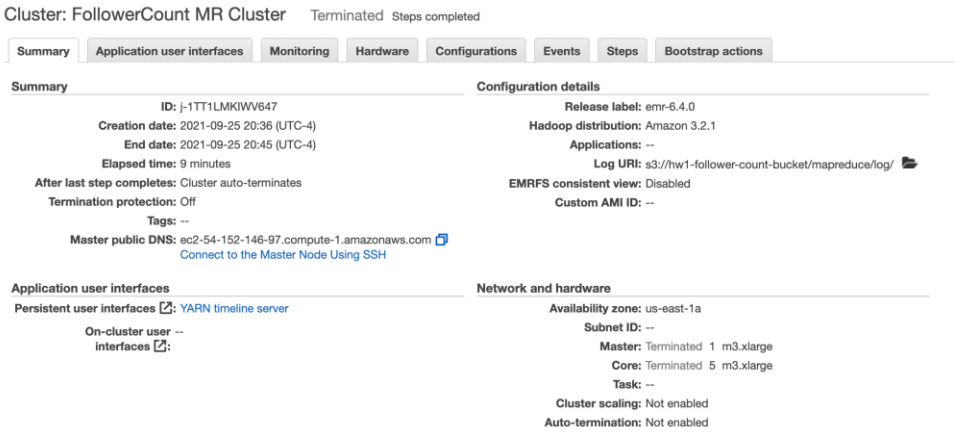
5. Pseudo-code for the Twitter-follower-count program in Spark Scala

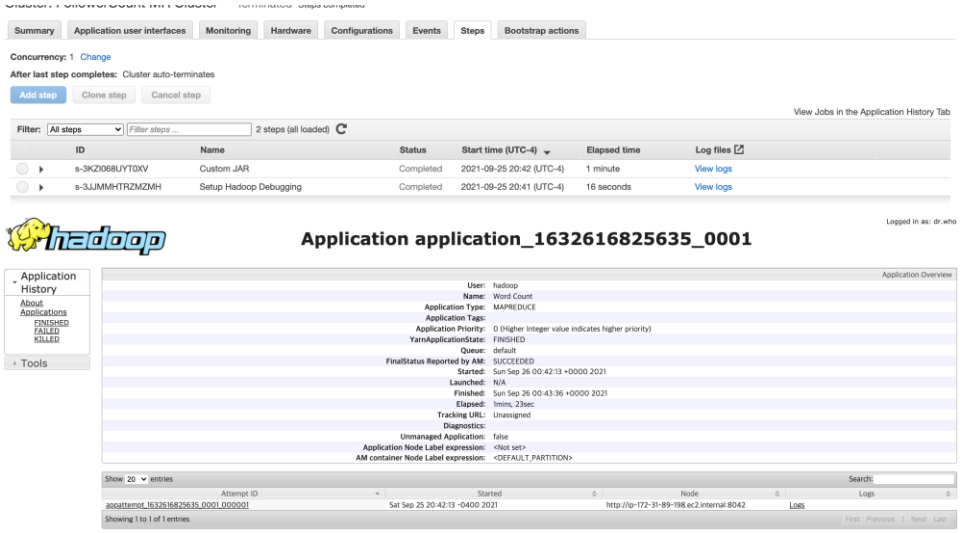
```
val textFile = sc.textFile(args(0))  // args(0) = input path. read the input csv file.
val counts = textFile.flatMap(edges ⇒ edges.split("\n"))  // split the input by whitespace "\n" and flatten the split edges to create a single RDD.
             .map(edge ⇒ edge.split(",")(1))  // for each nodes, split by "," and get the second element which is the Twitter ID being followed.
             .map(userID ⇒ (userID, 1))  // map each of the userID to create the pair (userID, 1).
             .reduceByKey(_ + _)  // sum up the count for each correspondig userID.
             .filter{case (userID, count) ⇒ count % 100 == 0}  // only get the userID with number of followers that can be divisble by 100.
logger.info("counts log starts here")
logger.info(counts.toDebugString)
logger.info("counts log ends here")
counts.saveAsTextFile(args(1))  // args(1) = output path. write the results (userID, total followers count) as the text file.
```

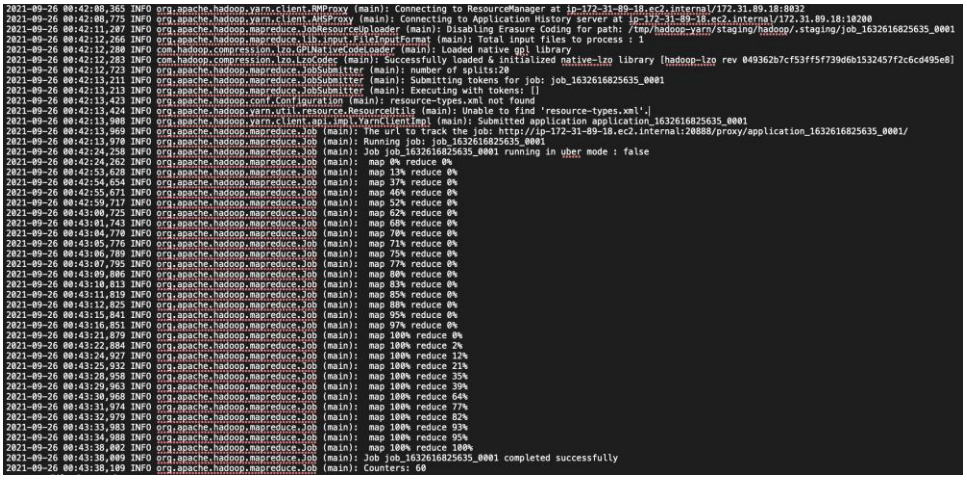
6. "counts" RDD Logger info

```
2021-09-25 15:19:08,104 INFO root: (40) MapPartitionsRDD[6] at filter at FollowerCount.scala:30 []
 |   ShuffledRDD[5] at reduceByKey at FollowerCount.scala:29 []
 +-(40) MapPartitionsRDD[4] at map at FollowerCount.scala:28 []
    |   MapPartitionsRDD[3] at map at FollowerCount.scala:27 []
    |   MapPartitionsRDD[2] at flatMap at FollowerCount.scala:26 []
    |   /Users/sunho/Dropbox/Boston/CS6240/HW1/twitter-dataset/data/edges.csv MapPartitionsRDD[1] at textFile at FollowerCount.scala:25 []
    |   /Users/sunho/Dropbox/Boston/CS6240/HW1/twitter-dataset/data/edges.csv HadoopRDD[0] at textFile at FollowerCount.scala:25 []
```

7. MapReduce and Spark programs on AWS
   a. MapReduce running time







Approximately took 1 minute 30 seconds.

b, c, d. Amount of data transferred

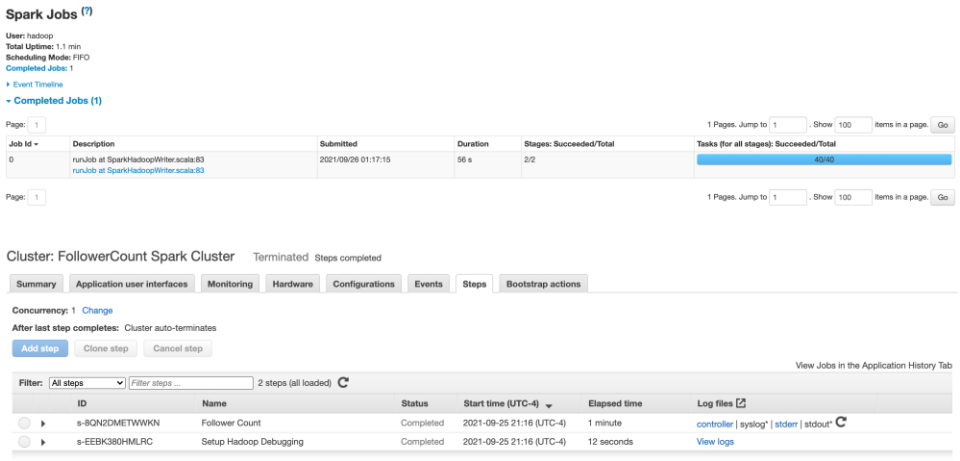
*based on syslog file.

```
File System Counters
        FILE: Number of bytes read=91879485
        FILE: Number of bytes written=245404466
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=2220
        HDFS: Number of bytes written=0
        HDFS: Number of read operations=20
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=0
        HDFS: Number of bytes read erasure-coded=0
        S3: Number of bytes read=1319425328
        S3: Number of bytes written=15294
        S3: Number of read operations=0
        S3: Number of large read operations=0
        S3: Number of write operations=0
```

```
Job Counters
         Killed reduce tasks=1
         Launched map tasks=20
         Launched reduce tasks=20
         Data-local map tasks=20
         Total time spent by all maps in occupied slots (ms)=39832830
         Total time spent by all reduces in occupied slots (ms)=39159270
         Total time spent by all map tasks (ms)=885174
         Total time spent by all reduce tasks (ms)=435103
         Total vcore-milliseconds taken by all map tasks=885174
         Total vcore-milliseconds taken by all reduce tasks=435103
         Total megabyte-milliseconds taken by all map tasks=1274650560
         Total megabyte-milliseconds taken by all reduce tasks=1253096640
Map-Reduce Framework
         Map input records=85331845
         Map output records=85331845
         Map output bytes=961483442
         Map output materialized bytes=144209432
         Input split bytes=2220
         Combine input records=0
         Combine output records=0
         Reduce input groups=6626985
         Reduce shuffle bytes=144209432
         Reduce input records=85331845
         Reduce output records=1339
         Spilled Records=170663690
         Shuffled Maps =380
         Failed Shuffles=0
         Merged Map outputs=380
         GC time elapsed (ms)=26645
         CPU time spent (ms)=698250
         Physical memory (bytes) snapshot=25376514048
         Virtual memory (bytes) snapshot=141496213504
         Total committed heap usage (bytes)=23091740672
         Peak Map Physical memory (bytes)=1030352896
         Peak Map Virtual memory (bytes)=3066417152
         Peak Reduce Physical memory (bytes)=510246912
         Peak Reduce Virtual memory (bytes)=4294897664
```

```
File Input Format Counters
         Bytes Read=1319425328
File Output Format Counters
         Bytes Written=15294
```

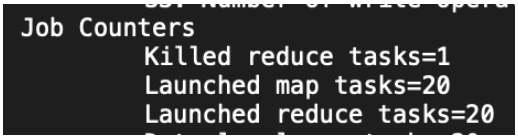
e. Spark running time



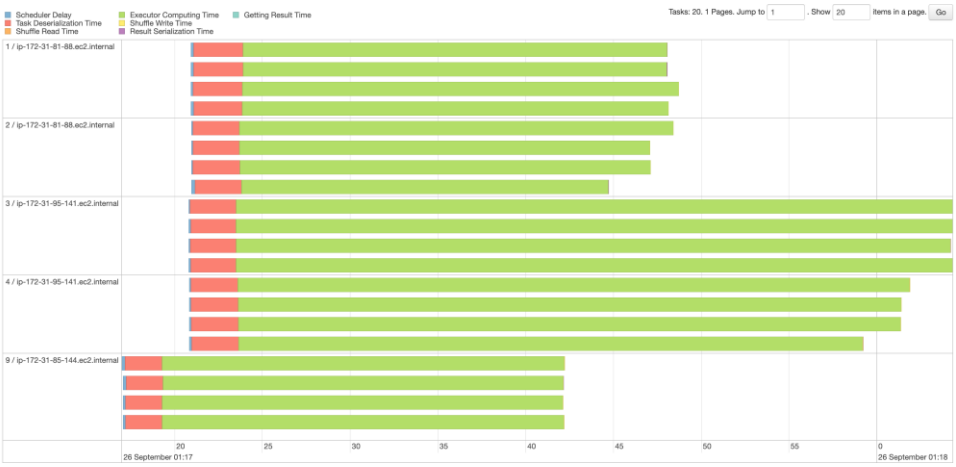
Approximately took 1 minutes.

8. Speedup discussion.

(1) Number of Map/Reduce tasks - 20 tasks



(2) Number of Spark tasks - 20 tasks



As shown from the figures above, Map/Reduce and Spark tasks both ran 20 tasks. Ideally having 20 workers working in parallel will improve the speedup.

9. AWS Log & Output Links

a. MapReduce syslog - hw-1-mapreduce-jill666666/aws-log at master · CS6240/hw-1-mapreduce-jill666666 (github.com)
b. MapReduce output - hw-1-mapreduce-jill666666/aws-output at master · CS6240/hw-1-mapreduce-jill666666 (github.com)
c. Spark stderr - hw-1-spark-jill666666/aws-log at master · CS6240/hw-1-spark-jill666666 (github.com)
d. Spark output - hw-1-spark-jill666666/aws-output at master · CS6240/hw-1-spark-jill666666 (github.com)