Jonathan Candelaria SID: 861062229 NetID: jcand003
Jesse Gomez SID: 861056174 NetID: jgome026
Yuxuan (Leo) Li SID: 861045931 NetID: yli066

CS179G: Bigdata Analysis Spring 2016

Project Phase 2 Report

Design and Implementation

We currently have three different mapreduce jobs which gives us the most popular hashtags and their total average sentiment, the most popular hashtag of each day and current average sentiment, and the location from which each of these popular daily hashtags are originating which will later be displayed on a heatmap.

We used Python to write each of these mapreduce jobs and implemented them with Hadoop Streaming. This allows for each mapper and reducer task to execute their own separate process, thus greatly increasing the speed at which data is processed. Using this streaming approach to Hadoop does affect performance as opposed to the regular Java implementation, however it does not seem to be significant according to the documentation.

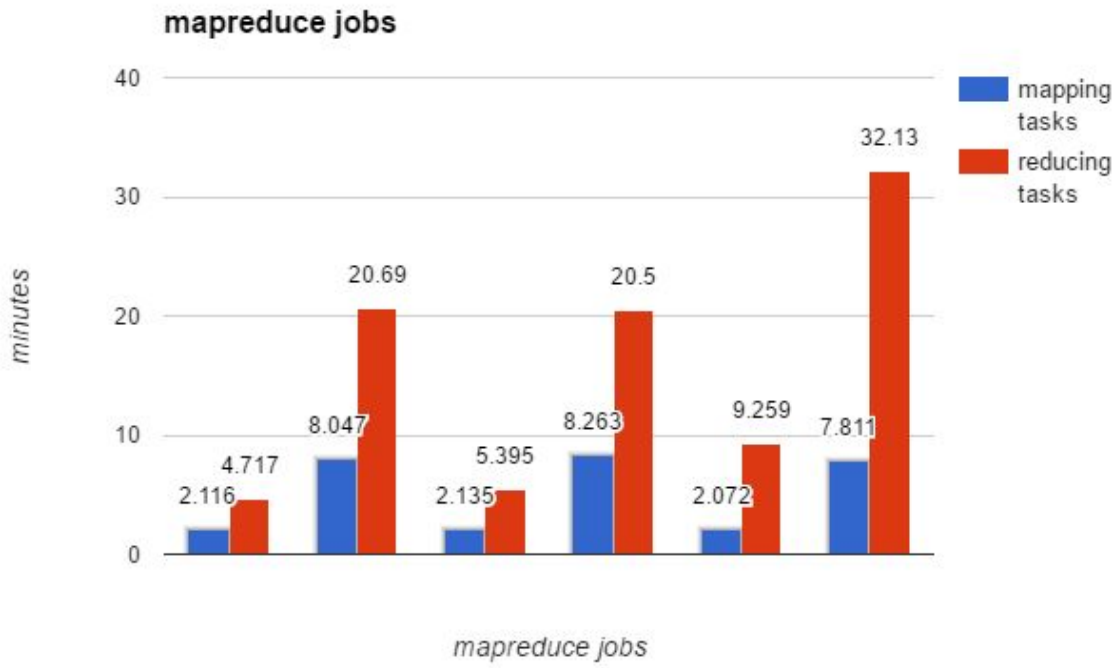
We also used a Python script to append the sentiment polarity and subjectivity of each tweet before performing our mapreduce. This is due to a problem we were unable to resolve with using the textual processing library in the Hadoop file system without being able to simply install it into Python.

Once our mapreduce jobs finished we transferred the output from the Hadoop filesystem to our local filesystem and executed our Cassandra script to automatically create a keyspace, create tables, and populate these tables with the data from the mapreduce output.

Evaluation

We downloaded a total of 40 million tweets over the period of about two weeks, which is approximately 7GB of data. We performed our three mapreduce jobs two times each, once with a fourth of the total data (10 million tweets) and once with all 40 million tweets. We also experimented with much smaller number of tweets to begin with and found that Hadoop was not as efficient with these smaller files. This is why we elected to concatenate all the tweets into one

large file once they were on the z7 servers, as opposed to feeding them into Hadoop in the 5 million chunks that we transferred them as.



Shown above are the timing results of the three mapreduce jobs, each with an input of 10 million and the with an input of 40 million. The difference in run times are close to linear between these two different data sizes as 10 million tweets is equivalent to about 1.6 GB which is a reasonable sized file for a Hadoop mapreduce jobs. However, if Hadoop tries to operate on multiple smaller files we begin to see drawbacks as it's not really worth the resources that Hadoop has to prepare to process this data.

Since we did not implement our own Hadoop cluster and instead used the one provided on the z servers, we were not able to conduct performance tests using a varying numbers of Hadoop nodes.

Member Contributions

- Jonathan: Wrote 2/6 of the python scripts.
- Jesse: Wrote 2/6 of the python scripts and the sentiment analysis script.
- Leo: Wrote 2/6 of the python scripts and the cassandra script.

Screenshots

Output from one of our mapreduce jobs

```
16/05/13 20:49:44 INFO mapreduce.Job: Job job_1462903890193_0473 completed successfully
16/05/13 20:49:44 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=515461546
        FILE: Number of bytes written=1038455649
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=7168189192
        HDFS: Number of bytes written=27808777
        HDFS: Number of read operations=165
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=54
        Launched reduce tasks=1
        Data-local map tasks=54
        Total time spent by all maps in occupied slots (ms)=1487403
        Total time spent by all reduces in occupied slots (ms)=3690204
        Total time spent by all map tasks (ms)=495801
        Total time spent by all reduce tasks (ms)=1230068
        Total vcore-seconds taken by all map tasks=495801
        Total vcore-seconds taken by all reduce tasks=1230068
        Total megabyte-seconds taken by all map tasks=761550336
        Total megabyte-seconds taken by all reduce tasks=1889384448
    Map-Reduce Framework
        Map input records=39369220
        Map output records=14507460
        Map output bytes=486446352
        Map output materialized bytes=515461864
        Input split bytes=5076
        Combine input records=0
        Combine output records=0
        Reduce input groups=1252063
        Reduce shuffle bytes=515461864
        Reduce input records=14507460
        Reduce output records=621006
        Spilled Records=29014920
        Shuffled Maps =54
        Failed Shuffles=0
        Merged Map outputs=54
        GC time elapsed (ms)=10308
        CPU time spent (ms)=1558640
        Physical memory (bytes) snapshot=64256671744
        Virtual memory (bytes) snapshot=121838718976
        Total committed heap usage (bytes)=67823468544
    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=7168184116
    File Output Format Counters
        Bytes Written=27808777
```

Example output from Cassandra table for total hashtags collected

```
cqlsh:csproj> SELECT * FROM tagranks WHERE cnt > 10000 AND pol < 1 LIMIT 10 ALLOW FILTERING;

 tag          | cnt     | pol      | sub
--------------+---------+----------+----------
       austin |   11508 |   0.0625 | 0.953125
 nfldraft2016 |   46932 |    1e-06 | 0.266169
      toronto |   16892 | 0.003906 | 0.368164
       retail |  192560 | 0.999951 | 0.890662
      traffic |   31056 |  0.98724 | 0.328971
 businessmgmt |   38060 | 0.990216 | 0.983947
        views |   96092 |    1e-06 | 0.758776
   realestate |   22132 |   0.0625 |  0.65625
          job | 1398696 |  1.5e-05 | 0.567374
      finance |   20132 | 0.999883 | 0.983496

(10 rows)
cqlsh:csproj>
```