Assignment 0

Vikram Bhatt * 1

Abstract

In this assignment we extracted temporal interaction graph properties for Twitter data(1 Terabyte accumulated from 17th October-16th December 2016) with Apache Spark in Scala(Zaharia et al., 2016).We looked into various statistics of the user tweets.

1. Frequency of Hashtags

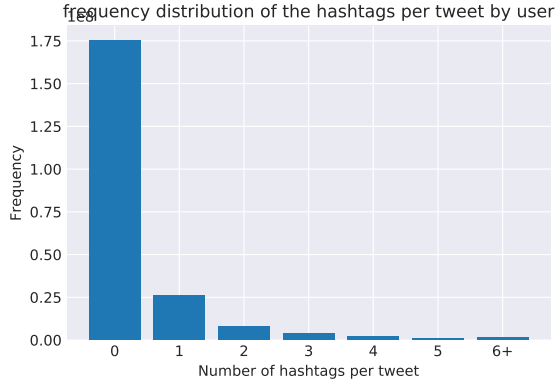


Figure 1. Frequency distribution of hashtags per user per tweets.

In question 1,we selected the sub-field of *entities.hashtags.text* of JSON object as prescribed by Twitter documentation, which is an array of type *Array[Row]* in Scala.We filtered out any null objects and mapped it to tuple of (*size of array of hashtags per user*, 1) and reduced by key with simple addition operation which gives (*Number of hashtags*,*Number of users used that many hashtags per tweet*).Fig.1 shows the bar plot of number of hashtags per tweet vs frequency.For the purpose of illustration,I have shown up to 5 and the rest are named into "6+" category.We have observed maximum value of 47 hashtags per tweet, but after 5 hashtags the frequency is extremely small resulting very skew distribution.We can see from 1 the majority of tweets don't use any hashtags.

Fig.2 shows the event timeline for the job for node1,node2,node3.As we can see,a hyper portion of timeline occupied by shuffle read time(orange) and executor compute time(green).We can also notice some idle time of nodes in between stages, maybe because data re-partition for the next stage and each node has to wait network bandwidth*data transferred units time.Shuffling is costly,and it has to be minimized for scalable algorithms.

Table 1. Various stats of the FreqTag Spark job.

PROPERTY -	VALUE ALLOCATED/OBSERVED
NUMBER OF EXECUTORS	4
DRIVER MEMORY	512MB
EXECUTOR MEMORY	8GB
EXECUTOR CORES	4
TOTAL TIME TAKEN	2.76HR

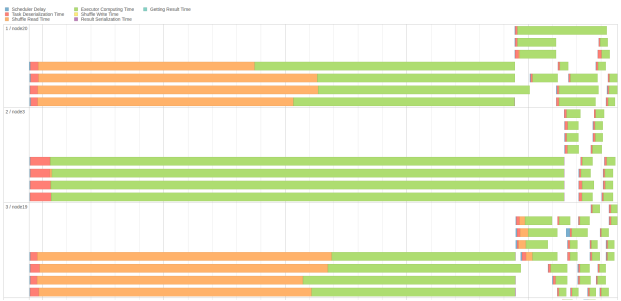
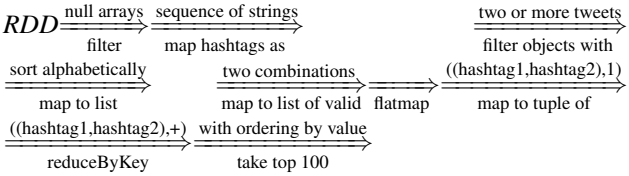


Figure 2. Events timeline taken for node1,node2 and node3 from turing.cds.iisc.ac.in:1801/history for various tasks performed in FreqTag job.

2. Top 100 Co-occurring hashtags

We selected *entities.hashtags.text* sub-field of JSON object and converted to RDD. Here the set of operations follows.



We have noticed (androidgames, gameinsight) is the most co-occurring tweet with frequency count of187356.Fig.3

Table 2. Various stats of the TopCoOccuring Spark job.

PROPERTY -	VALUE ALLOCATED/OBSERVED
NUMBER OF EXECUTORS	4
DRIVER MEMORY	512MB
EXECUTOR MEMORY	8GB
EXECUTOR CORES	4
TOTAL TIME TAKEN	4.1HR

shows the frequency distribution of the top 100 pair of hash-tags retrieved from the data.Fig.4 shows the event time line retrieved for the job, as we can see node1,node2 and node3 spent most of the time computing tasks and very few time spent on node3 for task shuffle read time which is close to ideal case.

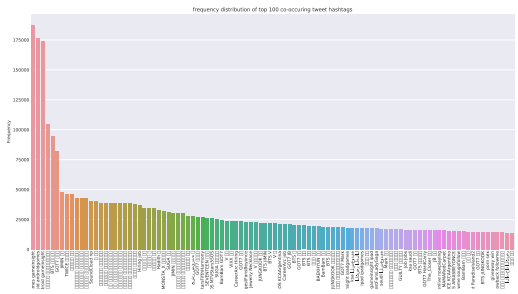


Figure 3. Top 100 co-occurring pair of tweets

Figure 4. Event timeline for the TopCoOccuring job.

3. Temporal Interaction Graph

We have extracted user.id_str,user.created_at,created_at, user.followers_count,user.friends_count sub-fields from JSON object and filtered out users with null values in first two fields and converted time stamps to epochs with Scala inbuilt functions.The following transformations follows. vertex RDD $\xrightarrow[\text{map to}]{\text{(userID,created\_at),(timestamp,followers\_count)}}$ $\xrightarrow[\text{reduceByKey}]{\text{(userID,created\_at),concatenate values}}$ and save to text file. Table 3 shows values set or observed for the spark job.

Table 3. Various stats of the InterGraph Spark job.

PROPERTY -	VALUE ALLOCATED/OBSERVED
NUMBER OF EXECUTORS	4
DRIVER MEMORY	512MB
EXECUTOR MEMORY	8GB
EXECUTOR CORES	4
TOTAL TIME TAKEN	11.2HR
VERTICES	39967256
EDGES	81555704

4. Time taken for 1%, 5%, 10%, 25% and 100% of job done

InterGraph job is divided into three stages(Table.4).We calculated by looking at the tasks need to be done for the given percentage.For example in order to be 1% of job to be done, we need to finish 414 task. Following this way,and using grep command on log files for files,(If I want to see time taken for 2068 tasks in stage 0 in logfile.txt,I would run **grep "task 2068.0 in stage 0.0" logfile.txt**).Fig.5 shows the graph for time taken in wall clock minutes extracted from log files.Table.5 shows the results.

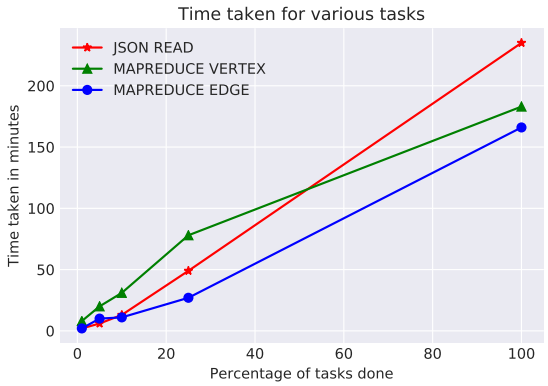


Figure 5. Time taken for various tasks in InterGraph job.

References

Karau, Holden, Konwinski, Andy, Wendell, Patrick, and Zaharia, Matei. *Learning Spark: Lightning-Fast Big Data Analytics*. O'Reilly Media, Inc., 1st edition, 2015. ISBN 1449358624, 9781449358624.

Zaharia, Matei, Xin, Reynold S., Wendell, Patrick, Das,

Table 4. Stages of InterGraph Job.

JOB	TIME	TASKS
PARSING JSON DATAA	2.9H	9406
MAPREDUCE JOB FOR VERTEX FILE	3.1H	7990
SAVE VERTEX TEXTFILE	12MIN	7990
MAPREDUCE JOB FOR EDGES FILE	2.8H	7990
SAVE EDGE FILE	9.2MIN	7990
		TOTAL=41366

Table 5. Time taken for different stages of job.

JOB	PERCENTAGE	TASKS FINISHED	TIME TAKEN
PARSE JSON	1%	94	2 MIN
	5%	470	6 MIN
	10%	941	13MIN
	25%	2352	49 MIN
	100%	9406	3 HR 55 MIN
MAPRED VERTEX	1%	78	8 MIN
	5%	400	20 MIN
	10%	799	31MIN
	25%	1998	1 HR 18 MIN
	100%	7990	3 HR 3MIN
MAPRED EDGE	1%	78	2 MIN
	5%	400	10 MIN
	10%	799	11 MIN
	25%	1998	27MIN
	100%	7990	2 HR 46 MIN

Tathagata, Armbrust, Michael, Dave, Ankur, Meng, Xiangrui, Rosen, Josh, Venkataraman, Shivaram, Franklin, Michael J., Ghodsi, Ali, Gonzalez, Joseph, Shenker, Scott, and Stoica, Ion. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, October 2016. ISSN 0001-0782. doi: 10.1145/2934664. URL http://doi.acm.org/10.1145/2934664.