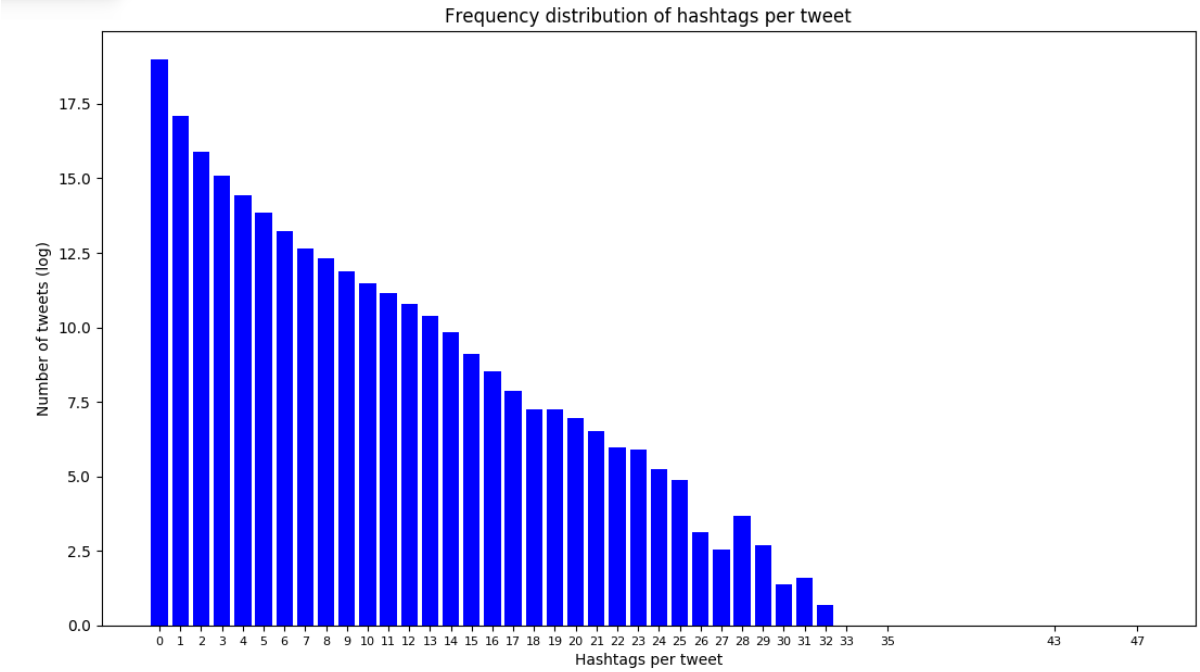
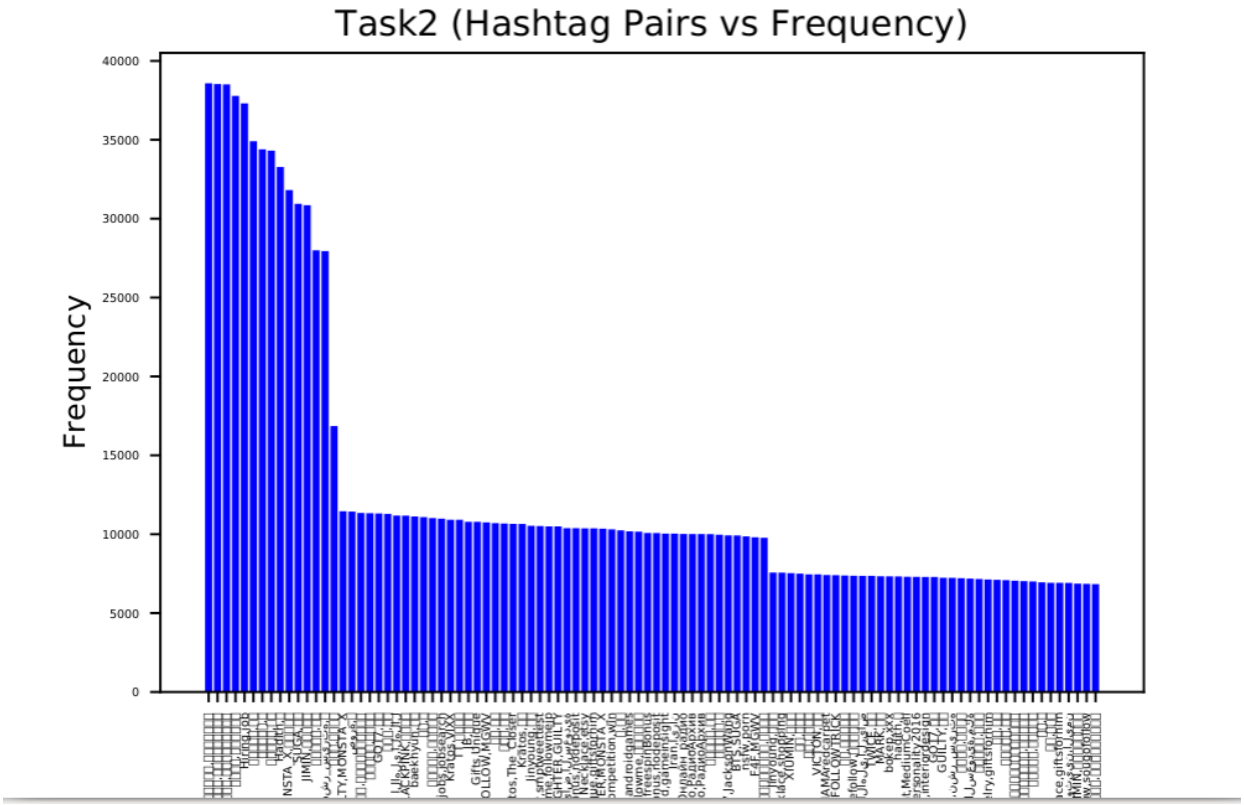
Task 1

1. Firstly, the input file is stored in an RDD of type String.
2. Now, each string in RDD is parsed using JSON parser and only the JSON objects are kept which have valid values to the keys we want such as hashtags. All other objects are filtered out.
3. Each element in the RDD is mapped to the number of hashtags it contains follwed by a mapToPair which pairs 1 to each RDD element.
4. Now, transformation reduceByKey is used to add the number of all the elements which contain same number of hashtags.
5. This RDD is saved using saveAsTextFile action.



Task 2

1. Input File is stored in an RDD of type string.
2. Now, each string in RDD is parsed using JSON parser and only the JSON objects are kept which have valid values to the keys we want such as hashtags. All other objects are filtered out.
3. All hashtags of a JSON object are stored in ArrayList of String and a HashSet is created to remove duplicates.
4. Now a new RDD is created containing pairs of HashTags present in each JSON object and mapToPair transformation is used to pair each element with 1.
5. Now, transformation reduceByKey is used to add the number of all the hashtag pairs.
6. This RDD is saved using saveAsTextFile action.



Top 100 Hashtag Pairs with their occurrences:

1. (38581,(つんつんしたい人あつまれ,ドッカンバトル))
2. (38535,(つんつくつん,アラレちゃんコラボ))
3. (38511,(つんつんしたい人あつまれ,アラレちゃんコラボ))
4. (37786,(モンスト,幽白モンスト))
5. (37306,(Hiring,job))
6. (34908,(방탄소년단,태형))
7. (34394,(방탄소년단,뷔))
8. (34311,(방탄소년단,지민))
9. (33274,(Hadith,□))
10. (31815,(MONSTA_X,몬스타엑스))
11. (30942,(SUGA,슈가))
12. (30852,(JIMIN,방탄소년단))
13. (27996,(방탄소년단,슈가))
14. (27946,(□,سيرة _نشر))
15. (16853,(갓세븐,하드캐리))
16. (11457,(GUILTY,MONSTA_X))
17. (11434,(□,صور))
18. (11345,(사설토토사이트추천,스포츠토토사이트추천))
19. (11330,(방탄소년단,피땀눈물))
20. (11315,(GOT7,영재))
21. (11290,(갓세븐,진영))
22. (11183,(الهلال,الاهلي))
23. (11180,(BLACKPINK,블랙핑크))
24. (11128,(baekhyun,백현))

25. (11084,(석진,진))
26. (11023,(강남야구장,선릉야구장))
27. (10983,(jobs,jobsearch))
28. (10918,(Kratos,VIXX))
29. (10916,(TT,트와이스))
30. (10786,(JB,갓세븐))
31. (10783,(Gifts,Unique))
32. (10749,(FOLLOW,MGWV))
33. (10704,(태태,태형))
34. (10675,(갓세븐,영재))
35. (10654,(Kratos,The_Closer))
36. (10647,(Kratos,빅스))
37. (10532,(Jinyoung,진영))
38. (10517,(ImWithHer,smptweettest))
39. (10497,(followme,followmejp))
40. (10497,(FIGHTER,GUILTY))
41. (10381,(الرياض,السعودية))
42. (10380,(casinobonus,nodeposit))
43. (10374,(Necklace,etsy))
44. (10372,(Unique,giftsforhim))
45. (10351,(FIGHTER,MONSTA_X))
46. (10308,(competition,win))
47. (10247,(뷔,태태))
48. (10182,(Android,androidgames))
49. (10160,(followme,相互フォロー))
50. (10078,(casinobonus,freespinBonus))
51. (10074,(freespinBonus,nodeposit))
52. (10039,(Android,gameinsight))
53. (10037,(Iran,ایران))
54. (10018,(Архив_радио,Онлайн_радио))
55. (10013,(Архив_радио,РадиоАрхив))
56. (10013,(Онлайн_радио,РадиоАрхив))
57. (10003,(몬스타엑스,원호))
58. (9966,(방탄소년단,진))
59. (9928,(GOT7,JacksonWang))
60. (9922,(BTS,SUGA))
61. (9863,(nsfw,porn))
62. (9805,(F4F,MGWV))
63. (9776,(RT した人全員フォローする,拡散希望))
64. (7571,(Jinyoung,갓세븐))
65. (7567,(Necklace,shopping))
66. (7533,(XIUMIN,시우민))
67. (7502,(붐붐,세븐틴))
68. (7457,(쯔위,트와이스))
69. (7452,(VICTON,빅톤))
70. (7411,(2016MAMA,MAMAredcarpet))
71. (7401,(FOLLOW,FOLLOWTRICK))
72. (7383,(길티,몬스타엑스))
73. (7365,(refollow,相互フォロー))
74. (7362,(الرياض,الاهلي))

75. (7360,(TWICE,미나))
76. (7332,(MARK,마크))
77. (7327,(bokep,xxx))
78. (7325,(hadith,?))
79. (7301,(Ascendant,MediumCoeli))
80. (7298,(ParthSamthaan,TVPersonality2016))
81. (7292,(decoration,interiordesign))
82. (7290,(GOT7,더쇼))
83. (7240,(GUILTY,길티))
84. (7235,(صورة,نشر_سيرته))
85. (7212,(とうらぶ,刀剣乱舞))
86. (7188,(السعودية,مكة))
87. (7159,(방탄소년단,전정국))
88. (7128,(Jewelry,giftsforhim))
89. (7110,(수호,준면))
90. (7089,(몬스타엑스,형원))
91. (7052,(RT した人全員フォローする,いいねした人全員フォローする))
92. (7025,(사설토토사이트추천,안전놀이터))
93. (7003,(그랜드게임,스위트게임))
94. (6954,(빅스,켄))
95. (6919,(個室,居酒屋))
96. (6918,(Necklace,giftsforhim))
97. (6914,(الحوثيين,اليمن))
98. (6867,(JIMIN,우리아미상받았네))
99. (6852,(refollow,sougofollow))
100. (6839,(메이저놀이터추천,사설토토추천사이트))

Task 3

VertexFile:

1. Input File is stored in an RDD of type string.
2. Now, each string in RDD is parsed using JSON parser and only the JSON objects are kept which have valid values to the keys we want such as user id, created at, timestamp, friends count and followers count. All other objects are filtered out.
3. Each JSON object is mapped to Pair RDD of type Tuple2 and Tuple3 where Tuple2 contains user id and created at (This created at is converted to Unix epoch time) and Tuple3 contains timestamp, friends count and followers count. This transformation is carried out on all the tweets.
4. GroupByKey transformation is carried out on the paired RDD having Tuple2 of user id and created at as key.
5. Now, each element of RDD is mapped to a string of desired ouptut(having desired delimitors) and saved using saveAsTextFile action.

No. of Vertices=39967256

EdgeFile:

1. Input File is stored in an RDD of type string.
2. Now, each string in RDD is parsed using JSON parser and only the JSON objects are kept which have valid values to the keys we want such as user id, retweeted_status, timestamp,

hashtags. Only the tweets which are retweets are taken(Tweets which don't have retweeted_status a s null).All other objects are filtered out.

3. Each JSON object is mapped to pair of Tuple2 and Tuple4. Tuple2 contains source_user_id,sink_user_id. Tuple4 contains timestamp,orig_tweet_id,retweet_id,hashtags.
4. GroupByKey transformation is carried out on the paired RDD having Tuple2 of source user id and sink user id as key.
5. Now, each element of RDD is mapped to a string of desired ouptut(having desired delimitors) and saved using saveAsTextFile action.

No. of Edges=81555704

HDFS:

Avg. block size = 113754955 B
Default replication factor = 2
Average block replication = 2.0
Number of data-nodes =23
Number of racks = 1

Cluster Setup:

No. of Nodes = 24
RAM = 47GB

Data Set Description:

No. of Files = 3984 (Out of which 1094 files are empty)
Each file contains approximately 1 lakh JSON Objects.
Each file size is approximately 300MB.
Each JSON object size is approximately 300KB.

Spark Environment Setup:

Memory Allocated = 8GB
No. of workers requested = 4
Driver memory =512MB

Task 4
(It was performed on 20 Jan 2019 because when I ran it on 18th Jan many other processes were getting executed on the cluster and most of the time was used for scheduling the processes and I got weird results)

1. 1% of data

a. No.of machines=4, Time taken to run=2.2 min
b. No.of machines=8, Time taken to run=1.7 min
c. No.of machines=12, Time taken to run=1.2 min

2. 5% of data

a. No.of machines=4, Time taken to run=7.5 min
b. No.of machines=8, Time taken to run=4.9 min

c. No.of machines=12, Time taken to run=4.5 min

3. 10% of data

a. No.of machines=4, Time taken to run=17 min
b. No.of machines=8, Time taken to run=11 min
c. No.of machines=12, Time taken to run=10 min

4. 25% of data

a. No.of machines=4, Time taken to run=41 min
b. No.of machines=8, Time taken to run=23 min
c. No.of machines=12, Time taken to run=23 min

As the number of machines are increased, the execution time is decreasing. Yes, this is matching my expectations.

Task 4 results when run previously:

No. of machines=4

1. 1% data Execution Time= 5 hours, appId=application_1547011148574_0367
2. 5% data Execution Time= 3.5 hours, appId=application_1547011148574_0368
3. 10% data Execution Time= 4.7 hours, appId=application_1547011148574_0369
4. 25% data Execution Time= 3 hours, appId=application_1547011148574_0370
5. 100% data Execution Time= 2 hours, appId=application_1547011148574_0219

These results did not match my expectations.