Project 4 – Part 1

Twitter Simulator using Elixir

Members

1: Manjary Modi ,UFID: 38408368, manjary.modi@ufl.edu
2: Rameshwari Obla Ravikumar,UFID: 16161302, rameshwari.oblar@ufl.edu

What is working?

For this project, we have created a twitter engine and a client test/simulator which has functionalities such as the ability to create multiple clients, send tweets from various clients, use of hashtag in tweets for easy collection and storing of data related to a particular topic, use of mentions in tweets in order to specify a certain user. A user can subscribe to other users by using the 'subscribe' option. This will allow users to get tweets directly from the subscribed user who he/she is following. Our Twitter engine supports 'retweeting' so as to allow sharing of interesting tweets. A user can also search for previous tweets by using the 'QuerySimulator' in our program. This will retrieve tweets from the data store of the engine and display the required tweets to the user querying the system. The query is also delivered to the client if the client process is alive.

Implementation Details

- *project4.ex*

This file contains the main method of our program. Based on the different input choice given by the user - server,client,live or query - different interface is created.

1. Server
Server which is an individual kernel process, creates the main server - our twitter engine. The server is a genserver node named as 'server@127.0.01' and it is also responsible for creating subservers. The use of subservers plays an important part and has been explained further in the document.

2. Client Simulator
This is also kernel process that creates multiple independent client process and connects them to the respective subservers which in turn are manged by the server created above.

3. Query
This is used to query the tweets based on hashtags(#), mentions(@) and subscriber information.

4. LiveQuery
This is used to display the tweets live into a client terminal if that client is alive. This will display the tweets from that user and all the tweets from the users whom the user has subscribed to.

- *mainserver.ex*

This file contain methods that performs the work of the main server or the central server. This simulates the Twitter Server.

1. Handle_cast method - ‘noOfClients
This method is used to create subservers, datacenters and hashtag centers. To support huge number of clients, the server is distributed. The mainserver creates several subservers based on ‘loadfactor’. The datacenters are also split based on the loadfactor. This is essential to improve the performance of our system.

2. Handle_cast method - ‘setUser’
This method is responsible for simulating ‘Registering an account’ by an user in our Twitter system. This method creates as many number of clients as specified in the program. This is responsible for creating users, linking the newly created client process to the server, to assign necessary states and to specify which subserver it reports to.

3. Handle_call method - ‘Tweet’
This method redirectes the tweet from clients to different subservers to process and store

4. Handle_cast method - ‘setFollowers’
This method is used to build the user-subscriber relation. It builds a list that maps every user to all the users to which he/she has subscribed to. This is used to query all the tweets tweeted by the subscribed users.

5. Handle_call method - ‘queryHashtag’
This method is used to query all the tweets with a given hashtag

6. Handle_call method - ‘getsubscribertweet’
This method uses the information created in the setFollowers method to get the tweets that is being queried for. ‘name’ refers to the user name or userid

- *server.ex*

This file contain methods that performs the work of the subservers. This is used to improve the performance of our system.

1. Handle_cast method - ‘initialize’
This method is used to initialize and fill the details of tweets in the datacenters, hashtag centers and the subserver.

2. Handle_cast method - ‘tweet’
This method is used in actual processing of tweets. It adds timestamp to the generated tweet, stores the tweets in appropriate hashtagcenter and datacenter. It also updates who subscribes to whom.

3.**Handle_cast method - ‘deliver_tweet’**
This method is used to display the tweets on the client process.

4. **Handle_cast method - ‘getsubscribertweet’**
This method is used to send subscriber tweets to the datacenter.

- *HashtagCenter.ex*

This file is used to create a hashtag center which can hold inforamtion about the hashtags that appear in each tweet. It groups all the tweet having a particular hashtag so that it can be returned while querying

1. Handle_cast method - ‘link_tweet’
This method is used to group all the tweets with a particular hashtag into a list of maps.

2. Handle_cast method - ‘gethashtag’
This method is used to get all the tweets containing the given hashtag.

- *datacenter.ex*

This file is used for storing subscriber and ‘mentions’ information. Mainly used in querying.

1. Handle_cast method - ‘store_tweet’
This method is used to store tweets with tweet_id, tweeter info and a flag to check if it is a tweet or a retweet.

2. Handle_cast method - ‘mentioned’
This method is used to store all the tweets in which a user is mentioned.

- *client.ex*

This file contains methods that performs client functionalities.

1. Handle_cast method - ‘startFollowers’
This method is used to create follower list. The followers are created by randomly picking variable amounts of followers for each user.

2. Handle_cast method - ‘tweet’
This method is used to generate and send the tweet. It also defines the structure of the hashtags and mentions.

3. Handle_info method - ‘startTweeting’
This method is used to send periodic tweets. More the followers, more the frequency of generating tweets. This facilitates the zipf distribution formlation in sending the tweets and is achieved by using rank of the user.

4. Handle_cast method - ‘subscriberTweeted’
This method is used to display tweets of the user and the subscribed users. Live Simulation is called from this method.

- *querysimulator.ex*

This file contains methods that are used in querying of tweets by mentions, hashtag and subscriber. All the methods are self-explantory

- *livesimulator.ex*

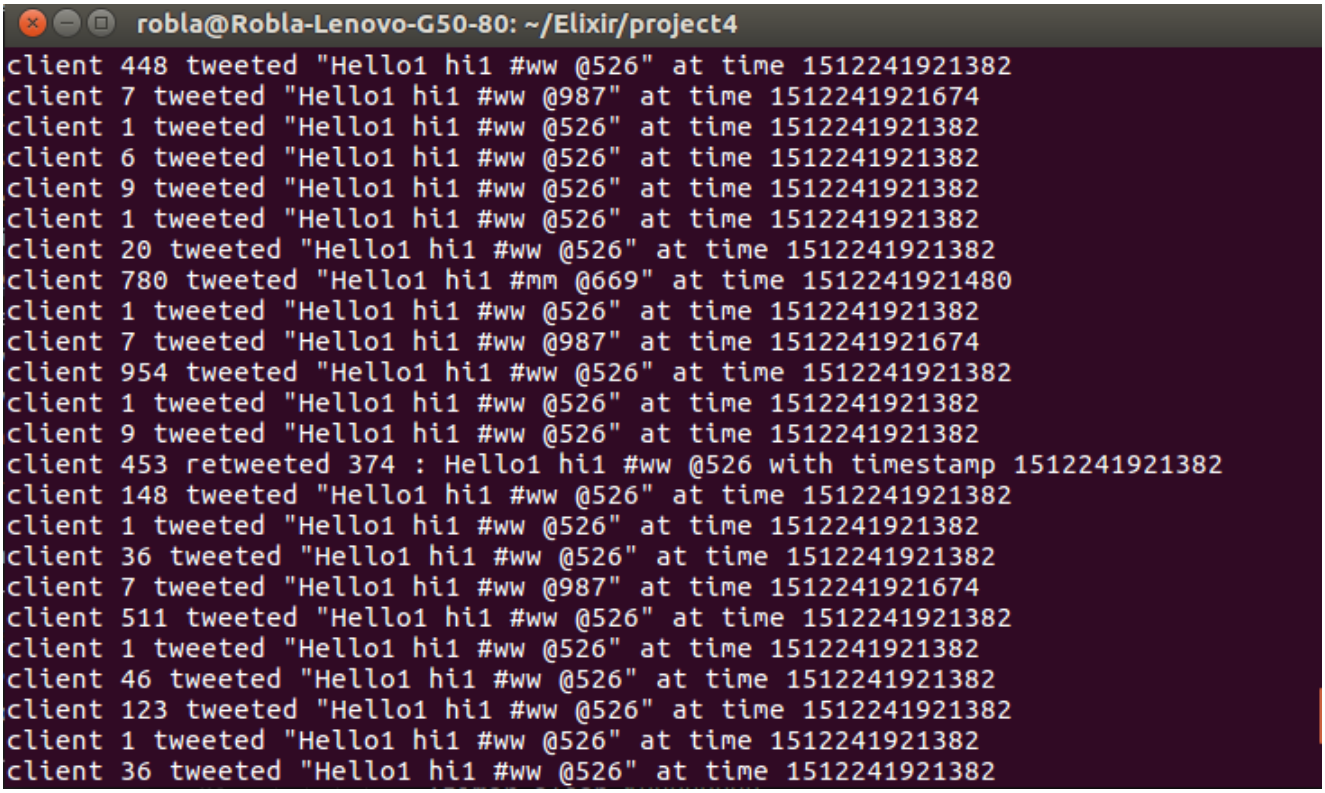
This file contains methods that displays subscribed tweets live on the user when it is alive.

1. Handle_cast method - ‘subscriberTweet’
This method displays the tweet from the user and the tweets tweeted by it subscribers.

Simulation Results

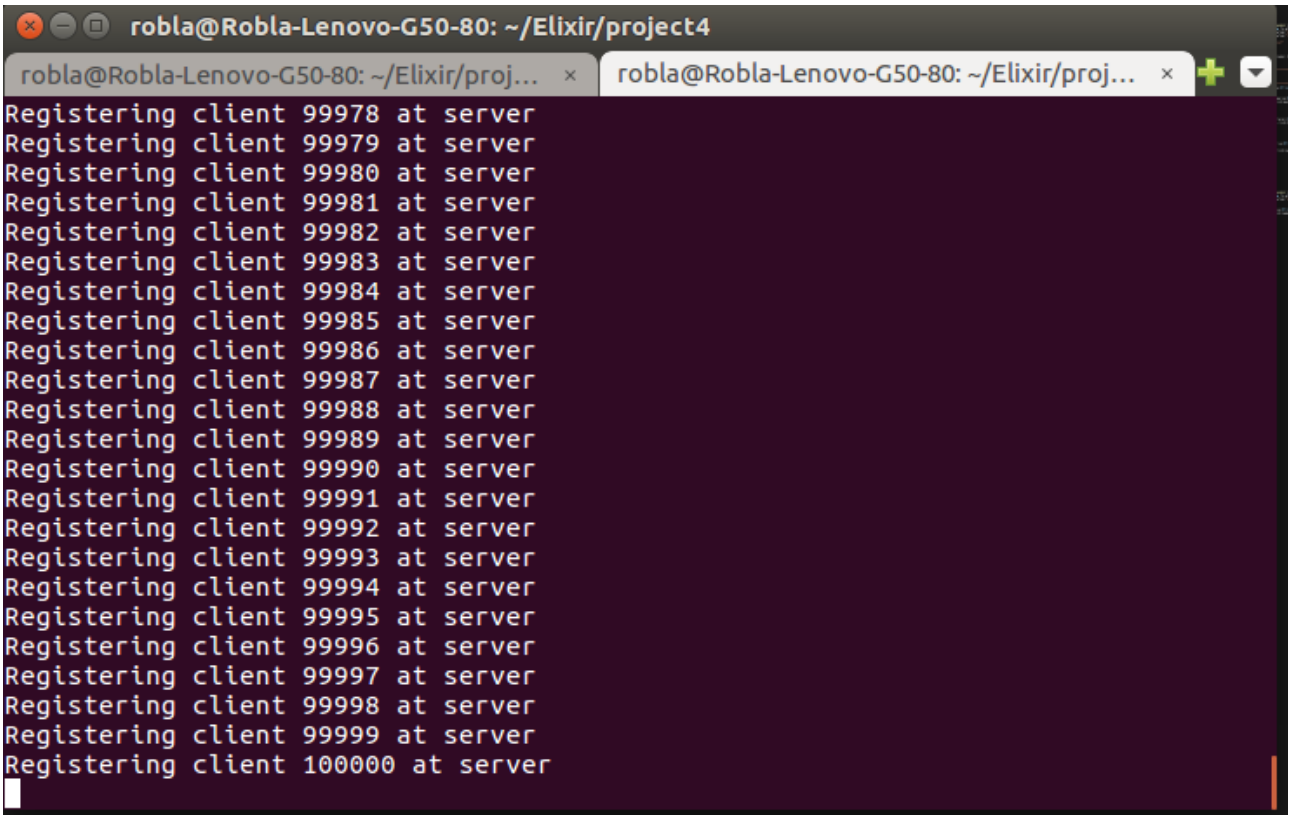
1. Tweets with Hashtag, Mentions and Retweeting.

Hashtags are prefixed by ‘#’ and Mentions are prefixed by ‘@’
Retweeting has information about original user who created the tweet.



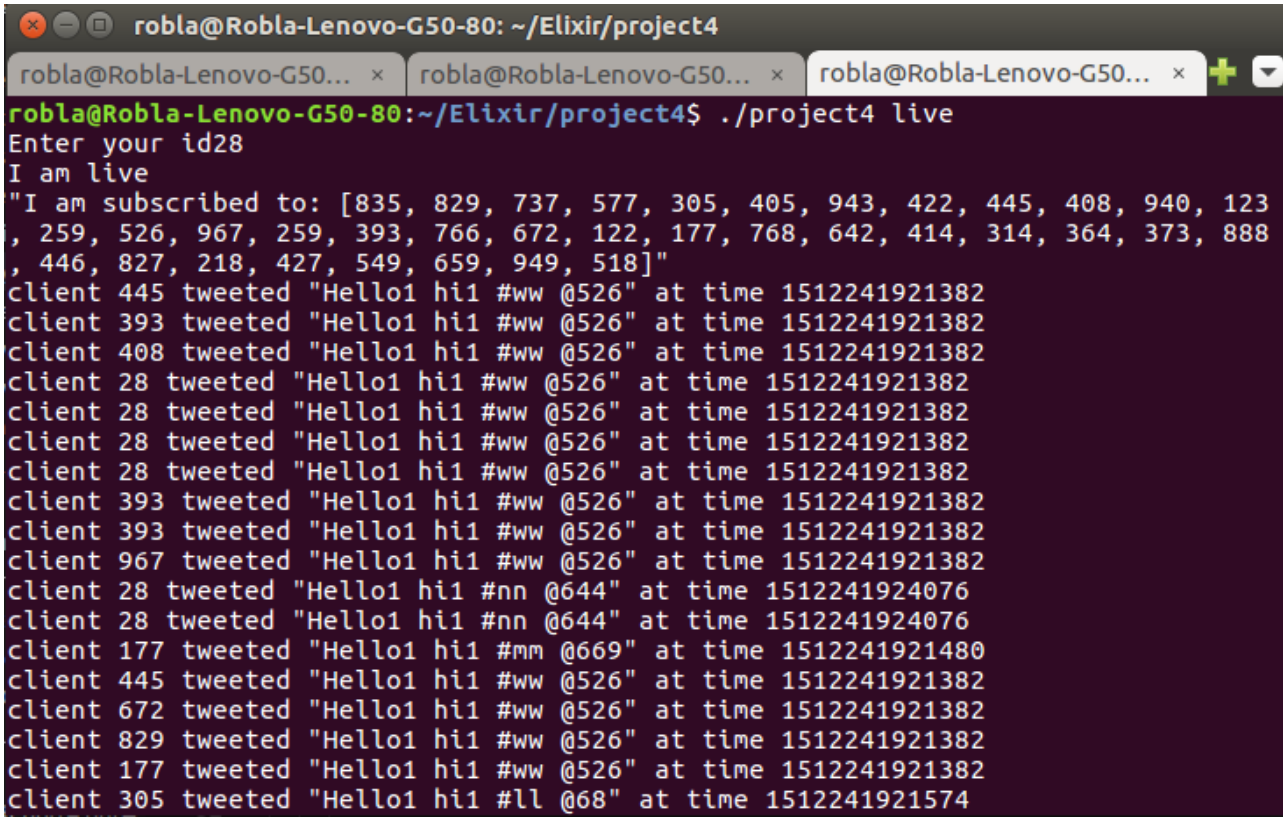
2. Max. No.of Users

We were able to simulate 100,000 clients in our system. It was using full 4 core power(Tested in a quadcore system). We believe more users can be simulated if more powerful CPU is used.
(The below screenshot also shows the registration process.)



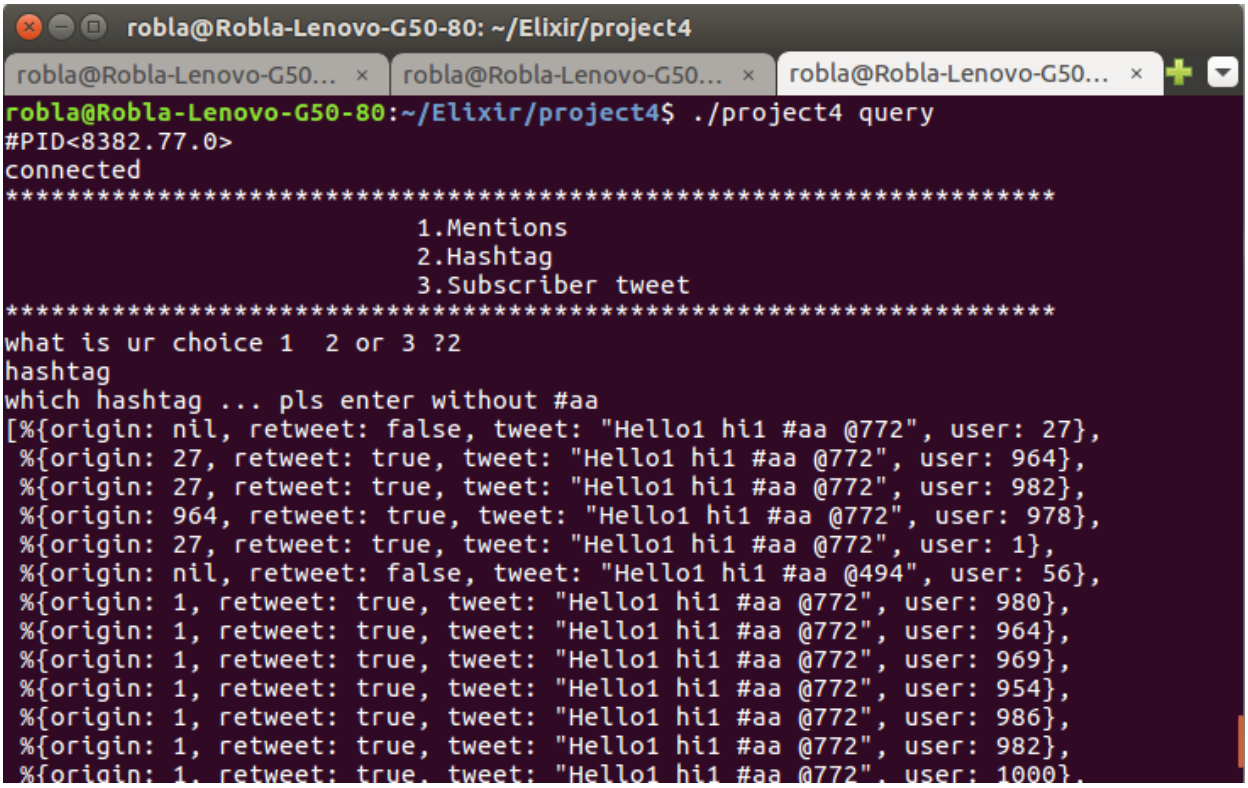
3. Live Tweeting(without querying)

We were able to simulate live tweeting by displaying the tweets of the user and all the tweets tweeted by users whom the user is subscribed to.

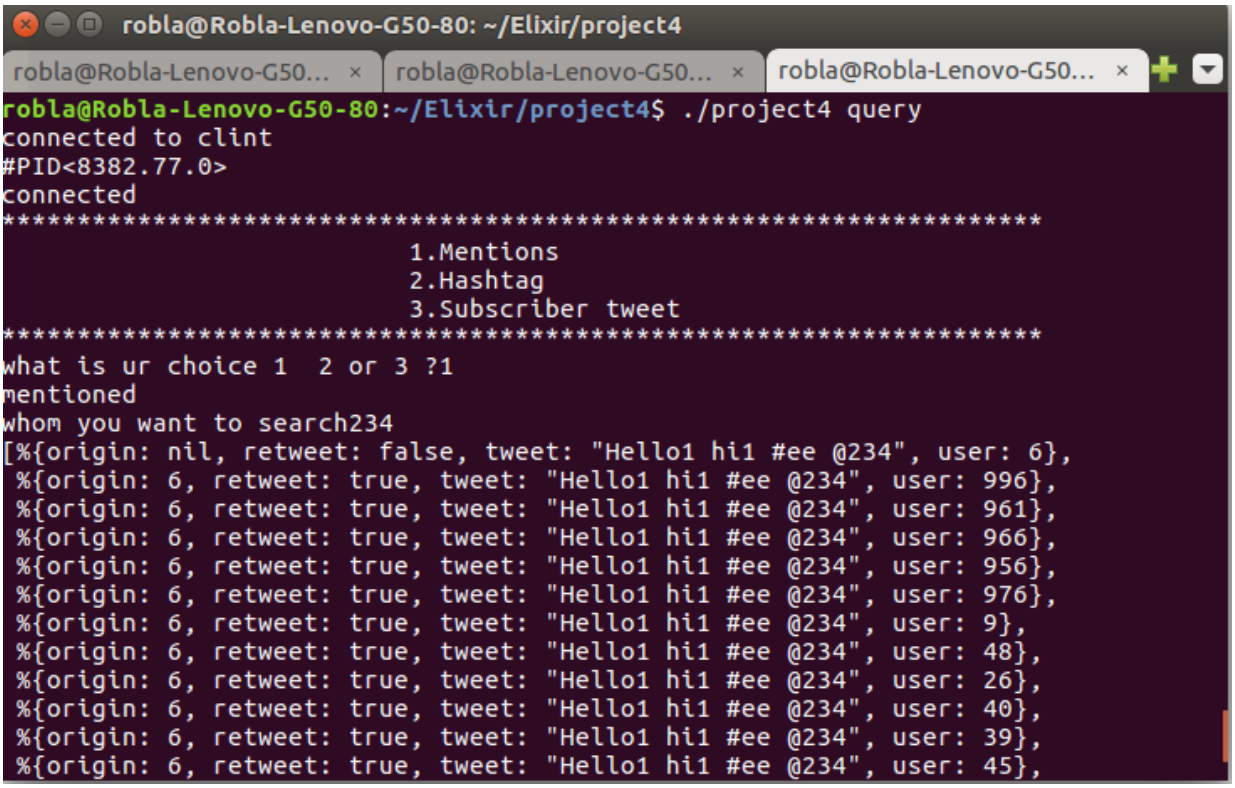


4. Querying of tweets by hashtag/mentions or subscriber.

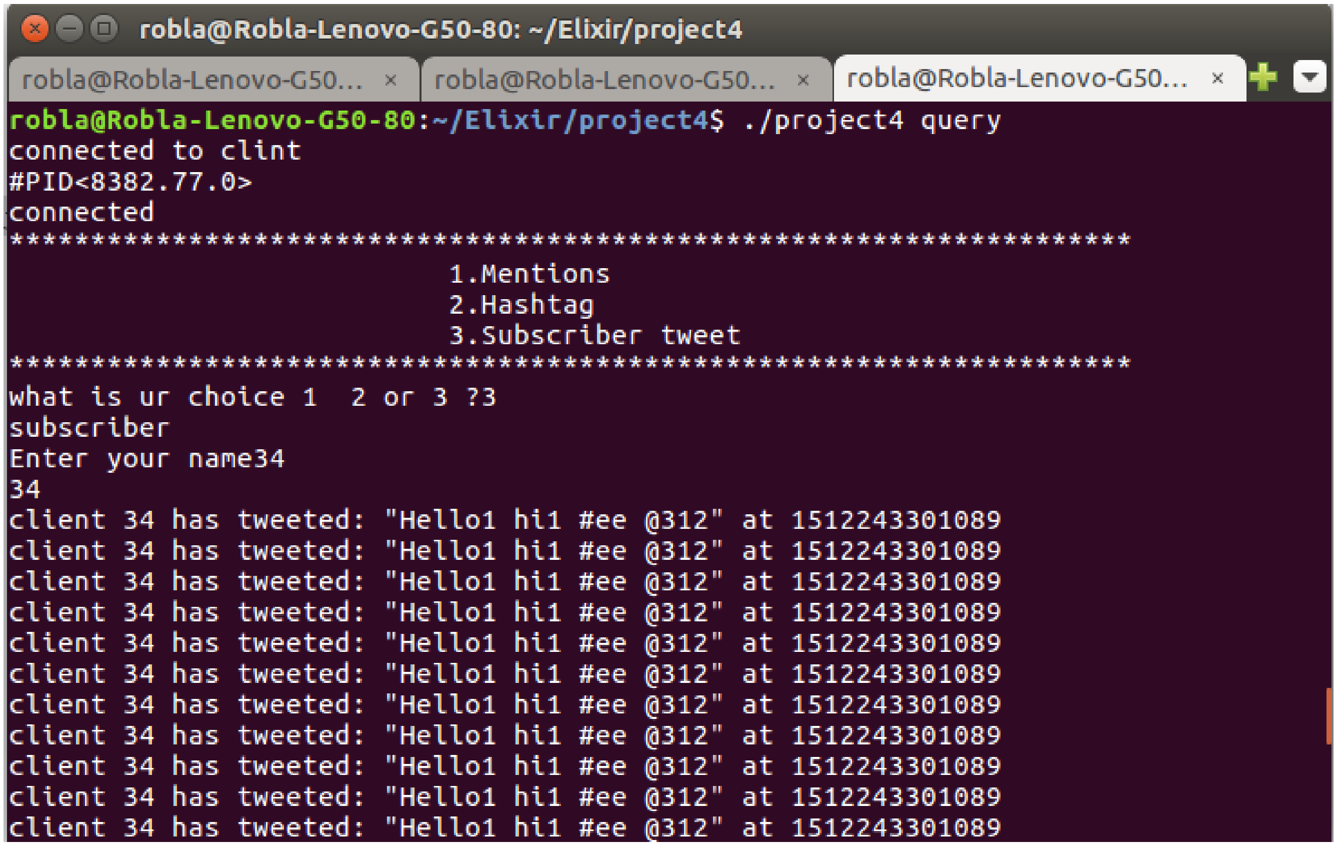
#Hashtag Querying



@Mentions Querying



Subscriber Querying



Performance Measurements

1. No.of users vs No. of tweets

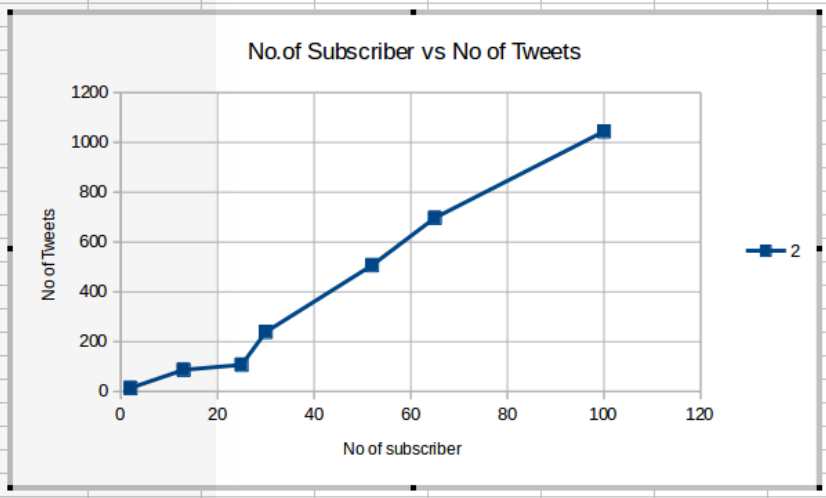
Then number of tweets is showing an increasing trend when the number of users are increasing. This is because, more the number of user, the tweeting will be done more frequent. But there is a bottleneck that when the number of user increases beyond a certain threshold, there will be to much load in the server and hence the server will not be able to process tweets with the same frequency as before. This can be seen from the below table:

No of Users	No Of Tweets/second
100	4916
500	5903
1000	6183
5000	6483
10000	4756
15000	3962
20000	3348

2. No. of subscriber vs No. of tweets

This simulates a zipf distribution on the number of subscribers. More the number of subscribers or followers, more is the number of tweets tweeted.

No of followers	No of Tweets
1	2
2	14
13	87
25	107
30	239
52	507
65	697
100	1044




```
Client 29 has 3 subscribers , tweeted 30 so far
Client 4 has 25 subscribers , tweeted 208 so far
Client 21 has 5 subscribers , tweeted 42 so far
Client 5 has 20 subscribers , tweeted 8 so far
Client 79 has 1 subscribers , tweeted 10 so far
Client 8 has 13 subscribers , tweeted 85 so far
Client 4 has 25 subscribers , tweeted 209 so far
Client 23 has 4 subscribers , tweeted 6 so far
Client 4 has 25 subscribers , tweeted 210 so far
Client 9 has 11 subscribers , tweeted 54 so far
Client 91 has 1 subscribers , tweeted 9 so far
Client 4 has 25 subscribers , tweeted 211 so far
Client 4 has 25 subscribers , tweeted 212 so far
Client 4 has 25 subscribers , tweeted 213 so far
Client 46 has 2 subscribers , tweeted 19 so far
Client 48 has 2 subscribers , tweeted 9 so far
Client 21 has 5 subscribers , tweeted 43 so far
Client 19 has 5 subscribers , tweeted 33 so far
Client 29 has 3 subscribers , tweeted 31 so far
Client 63 has 2 subscribers , tweeted 14 so far
Client 41 has 2 subscribers , tweeted 5 so far
Client 8 has 13 subscribers , tweeted 86 so far
Client 4 has 25 subscribers , tweeted 214 so far
Client 4 has 25 subscribers , tweeted 215 so far
Client 69 has 1 subscribers , tweeted 14 so far
Client 4 has 25 subscribers , tweeted 216 so far
Client 38 has 3 subscribers , tweeted 5 so far
Client 75 has 1 subscribers , tweeted 2 so far
Client 4 has 25 subscribers , tweeted 217 so far
Client 80 has 1 subscribers , tweeted 12 so far
Client 21 has 5 subscribers , tweeted 44 so far
Client 4 has 25 subscribers , tweeted 218 so far
Client 8 has 13 subscribers , tweeted 87 so far
Client 4 has 25 subscribers , tweeted 219 so far
Client 29 has 3 subscribers , tweeted 32 so far
Client 4 has 25 subscribers , tweeted 220 so far
Client 64 has 2 subscribers , tweeted 13 so far
Client 4 has 25 subscribers , tweeted 221 so far
```