**Project 4 Part 1: Twitter Clone in Elixir**

UFID: 39372723 Sarah Grace Samji

UFID: 61619287 Jagan Mohan

We have written a program where we are designing and simulating the Twitter system. We have tested this code on a quad core machine.

**Project run commands:**

mix escript.build

./project4 <number of clients>

**Description**

* Register account
* Send tweet. Tweets can have hashtags and mentions
* Subscribe to user's tweets
* Re-tweets (so that your subscribers get an interesting tweet you got by other means)
* Allow querying tweets subscribed to, tweets with specific hashtags, tweets in which the user is mentioned (my mentions)
* If the user is connected, deliver the above types of tweets live (without querying)
* Simulate periods of live connection and disconnection for users
* Simulate a Zipf distribution on the number of subscribers. For accounts with a lot of subscribers, increase the number of tweets. Make some of these messages re-tweets

**Assumptions Made**

* The server and the client are on separate nodes
* The number of tweets per user is controlled by the timing of client tweet requests. This timing is based on the Zipf distribution of the subscribers.
* No of retweets are very low in comparison to the tweets send out
* We have maintained a static list of hashtags, mentions and tweets that are used as inputs to generate the tweets. Each tweet can contain 1 hashtag and 1 mention.
* To allow for convenient understanding of the output and preventing interleaving of results: timer.sleep() has been inserted in multiple locations. This value has been maintained during submission.

**Modules in the project**

* Server: This resides on a node and is where all the ETS tables reside. The server is the final and only point of contact with the datastore. All requests from client are forwarded to the server, processed and the corresponding tables are updated. Results are sent back

Tables used:

* register: [UserID, password, variable to track whether online]
* followers: [UserID, followers]
* user\_tweets: [UserID, tweets received directly]
* hashtags: [Hashtags, UserID, tweets]
* mentions: [mentions, UserID,Tweets]
* followee: [UserID, followee list]
* Client: The client sends query requests, tweet, retweet, show notifications to the server. All this is controlled by the simulator. Once a client is started it is registered with the server.
* Simulator: The simulator provides the test cases for the client to run. It includes the functions to create follower list, build zipf distribution, create followee list, send tweet, show tweet, query tweets, retweet, query hashtags, query mentions, and live and disconnection periods.

The zipf distribution was made with the subscriber list and all tweet timings are controlled by the zipf. This is a ranking of the probability distribution of tweets across all clients.

**Project performance**

* User+sys/real=3.4
* 100% utilization. All 4cores were utilized.

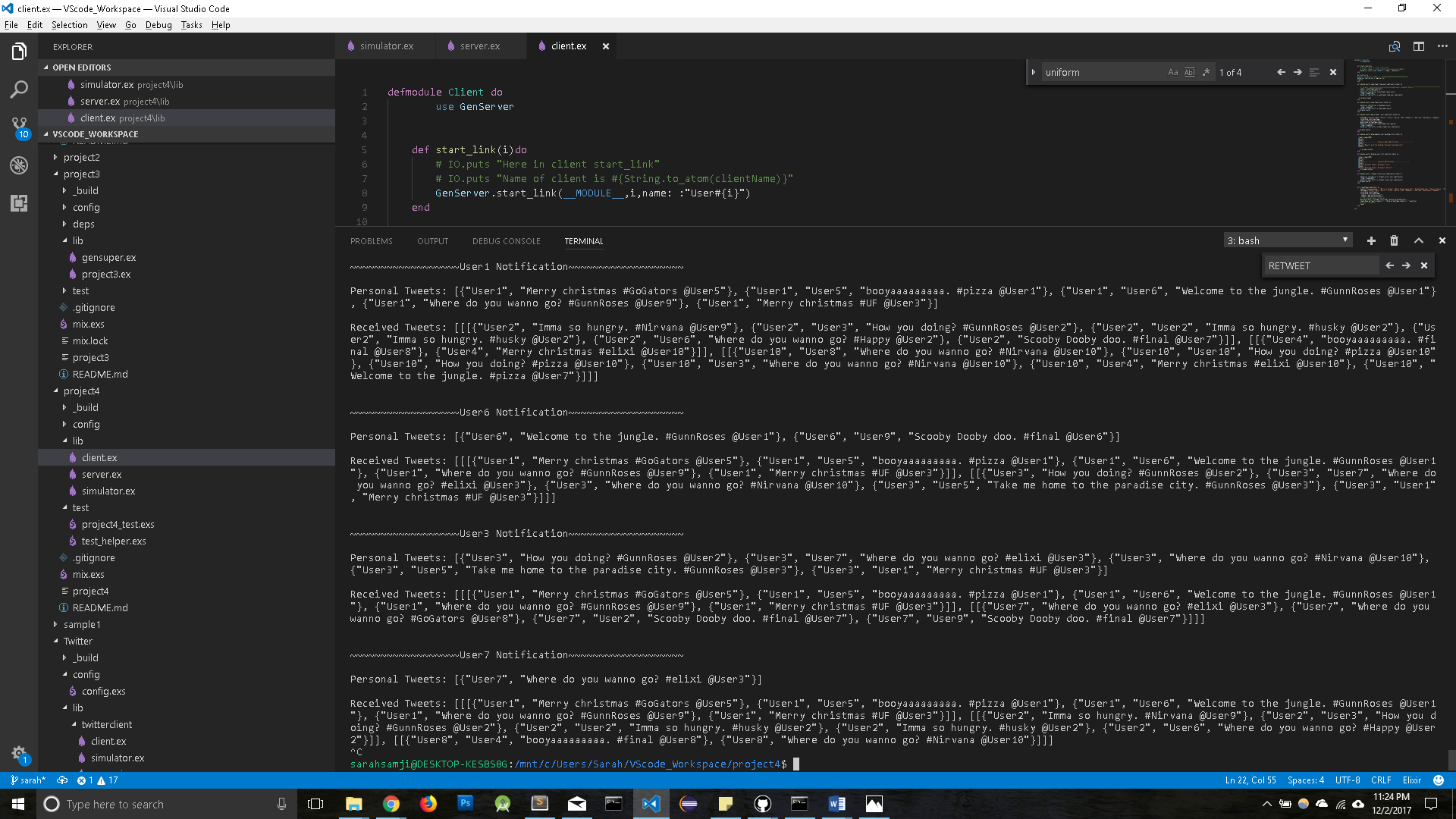
Snap showing Retweets:



Snap showing: The personal tweets and the received tweets

Personal tweets are those sent by the user himself

Received tweets are those that he receives from the followers.



**Results and conclusions**

• The average tweets/second observed for these many users kept on decreasing eventually.

• The reflected results are influenced by number of users, their tweeting frequency, the frequency at which the timeline is requested from the server and the network connecting the server-client machines.

Snap showing querys: Hashtags, mentions have been queried

