COP5615 – Distributed Operating Systems
Project-4

Manish Kumar Gangula (UFID:80727986)
Jaswanth Reddy Kankanala (UFID: 22719671)

In this project we are designing a twitter clone along with a simulator to test the scalable functionality for the extended users.

Twitter Engine:

Firstly, we need to design an engine which handles the requests from the users that involves things like signing in to twitter, registering a new account in twitter, once account is created a user can start tweeting about anything, mention other users in their tweets, use hashtags in their tweets, search tweets respective to their queries. The developed Twitter Engine uses Akka actor model to support the functionalities.

Twitter Simulator:

Similarly, the above discussed functionalities are basically simulated across varying number of users with the live connection and removing their connection to the twitter clone. We have achieved the same using the Zipf distribution. Also, part of the simulation we need to increase the number of tweets for the accounts with many subscribers to test the distributed messaging in twitter clone including some of these messages are re-tweeted from the other accounts.

Considerations:

Some other explicit aspects that the project includes are the separate process of both the above discussed functionality that can be useful in the process of scaling the functionality of increasing the tweets to be distributed and increasing the number of clients. We need to evaluate the performance of the system by evaluating the statistical parameters for understanding the performance of the design.

Implementation:

We have implemented the above problem statement considering twitter engine as our server and the simulator as the client.

1. **Server:**
   a. The server takes care of the twitter clone functionality like getting registered/ create new account, signing in, signing out, tweeting content, subscribing to other accounts, re-tweeting other user content, Querying/ searching content, tagging / mentioning other uses, adding hashtags, et.,
   b. We design the system in such a way that each of the above discussed service is given to an individual actor to handle it and provide the necessary response for the client's request.
   c. Master request handler that accepts all of the incoming requests and redirects to the respective actor which is assigned to it which serves a vital role in this system.
   d. Several properties of the user account are stored at the server level such as the number of tweets made, list of subscribers for the account, hashtags used, mentioned accounts in the tweets, re-tweets made such that they maintain the state of the data even after the account is signed out.
   e. All of the tweets that the account holders made are encrypted using the special symmetric encryption method such that the server and client both maintain the shared key to decrypt the content posted.

2. **Client:**
   a. The first and foremost task the simulator does is that it creates the accounts to validate and evaluate the functionality discussed above and also assess the performance of the design.
   b. In order to perform them, the client needs an account and can perform them after signing into the account. Nextly, we have a lot of operations at out plate, and we can perform them as like subscribing to some accounts, tweeting, retweeting, querying, and signing out in any of the order the client wishes to do so.
   c. Some of the pre-made tweets and freely available hashtags that user can use in the tweets. Every tweet that the client makes is already encrypted as discussed already. We can use multiple accounts to simulate the other users and perform the functionalities.

Zipf Distribution:

   a. In order to assess the performance of the system we need to add large number of clients and subscribe them to other account holders. In order to operate them we are using Zipf distribution which basically uses the ranking of the account.
   b. By storing the account references, we can rank the existing users, let's consider we have 'x' users that needs to be simulated. Now the user with the highest rank will tend to have 'x-1' users to be subscribed to, similarly we have 'x-1/2', 'x-1/3' for the subsequent rank holders.
   c. Each and every user is assigned with a time interval where they perform the functionality using the Zipf distribution.
   d. Which results in the account with the greatest number of subscribers will end up performing more of the functionality as discussed above(b) when compared with the less ranking account holder.

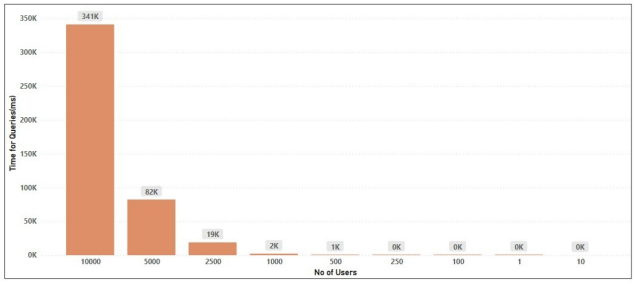
Commands to run the Project:

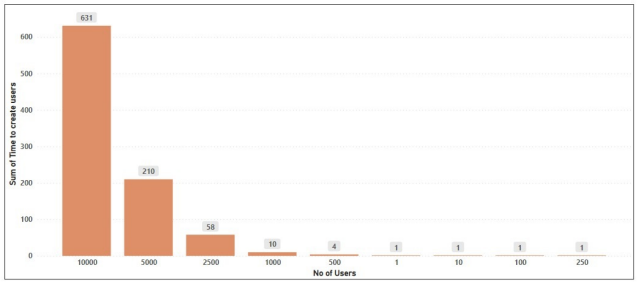
The design of the project includes two erlang files each one for the Twitter Engine(server) and the Simulator(client). Firstly, we need to compile both erlang files and then start the engine such that it accepts the requests from the users created using simulator with the commands below:

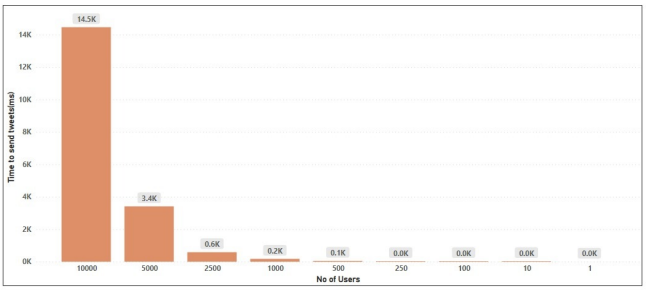
Compile the project: c(twitterengine). and c(twitterclient).
Run the Project: twitterengine:start(). And twitterclient:simulatetwitter(Number of Clients). {Number of Clients are varying and can take random values to assess the performance}.

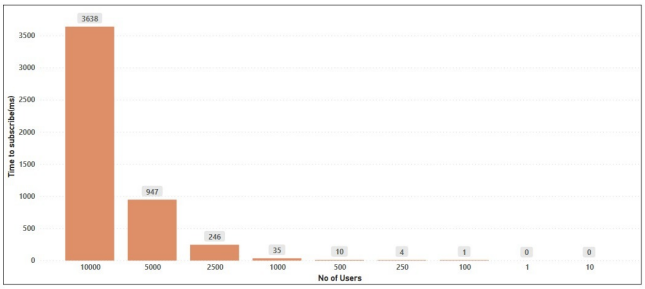
Zipf on Number of Subscribers:

# Users	Time took to create users(ms)	Time took to subscribe users(ms)	Time took send tweets to users(ms)	Time took to do Queries(ms)
1	**1**	**0**	**0**	**1**
10	**1**	**0**	**1**	**0**
100	**1**	**1**	**4**	**13**
250	**1**	**4**	**12**	**105**
500	**4**	**10**	**52**	**508**
1000	**10**	**35**	**174**	**2214**
2500	**58**	**246**	**586**	**18869**
5000	**210**	**947**	**3412**	**82299**
10000	**631**	**3638**	**14459**	**341244**









Zipf distribution plots with X-axis as the Number of users and Y-axis as the columns in the above-mentioned table.

Performance Evaluation:

The above Twitter clone project simulation is performed for 10000 users with client 1 simulating 3000 users and client 2 with 4000 users and client 3 with 3000 users.