Project Report: Twitter Simulator

Group Members

- Kalpak Seal – 8241 – 7219
- Sagnik Ghosh – 3363 – 6044

Functionalities

- Register Account: Based on the user input of numNodes, we are generating the users, we are choosing 5 random users for deletion and we are making these activated again.
- Send Tweet: From a random pool of tweets and hashtags, the users starts sending tweets mentioning other users and using hashtags.
- Subscriber: We are assigning subscribers to users, we have used ETS tables for storing subscriber of any user and on a separate table we store the users one account is subscribed to.
- Retweet: Users can retweet, the same is being logged on command line.
- Querying of Tweets: A new tweet is inserted into tweet database, which is implemented by ets tables. If any user is mentioned or a hashtag is used, the same tweet ID we are storing on the respective hashtag storage, "map_of_hashtag" and on mention storage "map_of_mentions". The same tweets can be queried by searching based on hashtags and listing "my_mentions".
- Receive Live Tweets: When user is online, we are delivering the tweets live, without querying.

Implementation

The client part and the server part has been implemented by separate processes, such that there would be only one server running and the clients (each of them, individual GenServer) will communicate with server.

The client part gets instantiated for each user and works independently. It sends or receives tweets. The server works as an engine and stores information and propagate tweets. We have implemented the actor model of Elixir, through GenServer. Through the state of the GenServer, we have kept track of tweets_seen, relevant tweets and etc.

Testing

We have unit tested the modules of our project. The Server has been tested for the functionalities,

1. **Get Tweet with hashtag** – We are inserting a tweet into our erlang storage, the server module inserts the tweet and corresponding hashtag. We queried tweets based on hashtags and we have passed the test case successfully.

2. **Mention Query -** We are inserting a tweet into our erlang storage, the server module inserts the tweet and corresponding mentions. We queried tweets based on mentions and we have passed the test case successfully.

3. **Test subscribe** – We tested whether subscriber is getting successfully added to subscriber list of any user.

4. **Test subscriber** – We tested whether a subscriber's subscribedto list is getting properly updated or not.

The unit test for the client module are:

1. **Hashtag parsing pass**– Based on a set of hashtag, this function returns truth value if a tweet has all the required hashtags.

2. **Hashtag parsing fail**- Based on a set of hashtag, this function returns truth if a tweet does not have all the required hashtags.

3. **Mention parsing pass**– Based on a set of mention, this function returns truth value if a tweet has all the required mentions.

4. **Mention parsing fail**- Based on a set of mention, this function returns truth if a tweet does not have all the required mentions.

Performance

Functionality	Time
Average Time Per Tweet	8.3 millisecond
Average Time for query tweets subscribed to	1.9 millisecond
Average Time for query tweets based on hashtags	3.1 millisecond
Average Time for query tweets based on mention	.9 millisecond
Average Time for query all relevant tweets	.6 millisecond

Observation

The project runs perfectly for 50000 users in a four core CPU. The simulation that we tested, the total number of tweets reported per second was 732 for 6000 users.