Twitter Clone and a Tester/Simulator

Project Execution Instructions:

To run the simulator:

- **mix escript.build**
- **./twitter server numUsers**

Where **numUsers** represents the number of twitter users

Commands To connect Users from different terminals/ machines: Open new terminal window:

- **./twitter client 'serverip' automatic**: This command will register and login a user node and connect it to the engine. It will also select two random online users from the engine and add the new user to their subscribers list
- **./twitter client 'serverip' register username**: This command will register a new user with the given username
- **./twitter client 'serverip' login username**: This command will login the registered user
- **./twitter client 'serverip' logout username**: This command will logout the registered user
- **./twitter client'serverip' follow follower_username following_username** : This command will add follower_username to the subscriber's list of following_username
- **./twitter client 'serverip' send_tweet 'username' "tweet":** This command will send tweet to the followers of provided username
- **./twitter client 'serverip' search_hashtag #hashtag**: This command will search the tweet with the provided hashtag
- **./twitter client 'serverip' search_usertweet @username**: This command will search the tweet which has tagged the provided username

Where **serverip** represents the ip address of the Engine/server

Implementation Details:

Twitter

We have implemented a twitter like engine with the following functionalities:

1. **Register Account**: As soon as the simulator starts, it registers all the users into the ETS database with incremental user ids. The username is stored as "**userid@localhost**"
2. **Authentication**: Apart from the user ids and usernames, **passwords and session ids** are also stored in the ETS database. Passwords are used as an authentication parameter to authenticate the user and refuse the connection if the wrong password is entered. Session ids are used to keep track of the sessions initiated by a user.

3. **Send-Tweet:** This functionality sends tweet to all the followers of a user. It also saves the tweet in the ETS database with username as the key. It also extracts hashtags and user mentions from the tweet and store them in different tables.
4. **Re-tweet:** A user can retweet a tweet and all its followers will receive that tweet. The retweeted tweets will have the owner username appended with it to identify the **username who initiated the tweet.**
5. **Hashtags & User Mentions Query:** The user can query for tweets with specific hashtags and user mentions. To implement the **constant lookup time**, the hashtags are stored as a map with hashtag as the key and a tuple with all the tweets that contains that hashtag as the value. The user mentions are stored similarly.
6. **User Online/Offline:** The user ETS table is also storing the online/offline status of the user. The user will **only receive the tweet if its online**.

Simulator:

1. **Users:** The simulator will automatically register all the users and the store their information in the database.
2. **Period of Offline/Online:** The simulator also simulates the online and offline period of users. The user will go in offline mode after sending some tweets and will again come online. The user in offline mode will not be able to perform any operation on engine. The period of offline/sleep will be randomly selected for different users.
3. **Zipf Distribution for Followers**: The simulator follows zipf distribution scheme to create subscribers for each user. The zipf distribution will make sure that some of the users are celebrity users and have large number of followers, while most of the users are normal users with very less number of followers.
4. **Offline/Online Period**: The offline/sleep period will be lesser for the celebrity users as compared to normal users. Hence most of the tweets will be send by celebrity users. The simulator also retweets some of the tweets initiated by the celebrity users.

Stress Testing:

To stress test our engine we have used the following schemes:

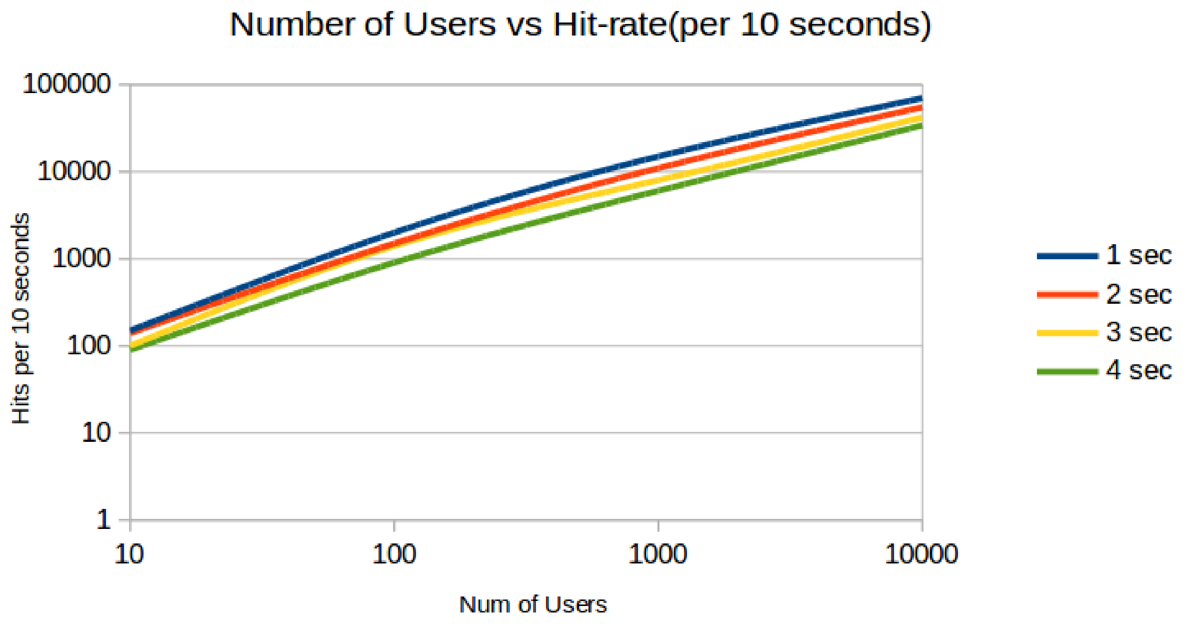
1. **Increase the number of Users:** The maximum number of users that we have tested our engine on are 100000. The zipf distribution takes approximately 15-20 minutes to register the users and create followers list.
2. **Join Users/Clients from different machines using Node.Connect:** The users can also query the engine from other machines and can successfully perform all the functionalities.
3. **Increase Tweet Rate/ Decrease Celebrity Users Sleep time:** Since most of the tweets are send by the celebrity users which have more number of followers, most of the stress will be created by

them. To test the robustness of our engine we have decreased the sleep/offline period such users so that the engine will receive more number of queries on a certain time interval.

Performance Observation:

Hit Counter: We have implemented a Hit Counter that will record the number of hits or queries per 10 seconds received by the engine.

- The below graphs show the number of user's vs the number of hits per 10 seconds when different tweet rates are selected.
- Line 1(blue): Offline/Sleep period in seconds for celebrity users before they come online and start sending tweets again: 1sec
- Line 2(Red): Offline/Sleep period in seconds for celebrity users before they come online and start sending tweets again: 2sec
- Line 3(Yellow): Offline/Sleep period in seconds for celebrity users before they come online and start sending tweets again: 3sec
- Line 4(Green): Offline/Sleep period in seconds for celebrity users before they come online and start sending tweets again: 4sec



Observation:

For less number of users, most of the twitter engine is underutilized and hence the hit rate remains the same irrespective of the Offline period. However, when the number of users increases the hit rate increases when we decrease the offline period of the celebrity users showing that most of the hits are performed by the celebrity users to the engine.