Rahul Bhatia 5993-0121
Foram Nirmal 6219-9417

REPORT
PROJECT 4.1

Project Description:

The goal of this was to implement a Twitter-like engine.

In our simulation our program starts off with the main server and a dynamic supervisor for each client.

- The main server creates the required number of clients by calling the dynamic supervisor as per the input given.
- All information is stored on ETS tables. We have tables for users, tweets, retweets, hashtags, mentions and subscribers.
- Each client is created with a status of logged in/logged out chosen randomly. They are also given the maximum number of tweets they must generate.
- The main server sends a message to all clients to start tweeting.
- Each client generates a random tweet by generating random text and picking out a hashtag randomly from the predefined list. Each tweet may or may not have a mention of another user.
- The tweets are sent to the main server which then checks who the tweet came from, picks out its subscribers from the ETS table and looks up the process ID and forwards the tweets to them.
- As each clients receives a tweet it adds the tweet to its twitter feed which is stored at each node.
- A tweet received may be randomly retweeted which then again follows the same path as a regular tweet, going through the main server.

We have run this simulation with up to 5000 clients with 100 tweets each. The program can be scaled endlessly as the logic is written to handle as many users and tweets needed as long as CPU power permits. As we add more users and tweets it would be better if we add more central servers to reduce run time as it bottlenecks on a single central server.

For the test scenario we have kept the simulation to 25 users with 10 tweets per user.
We have written various test functions to check correctness as well as show some stats to show every functionality works.

Test functions:

- Check if all ETS tables exist:
  We run tests and check whether all 6 tables exist.

- Check if the right number of accounts are created:
  We run a test to count all the inserted rows in the user table and match it with the number of clients we have.

Rahul Bhatia 5993-0121
Foram Nirmal 6219-9417

- Show logged in users:
  This is to show that some of the users are logged in and some are logged out. This is done on random and has a different result every time.

- Check if every user has at least 1 subscriber:
  We check to see if every user has at least 1 subscribed since we randomly pick them.

- Check if every user is not subscribed to itself:
  We run a test and check the subscribers of each user to make sure it is not subscribed to itself.

- Subscribers of each user:
  We print out the subscribers of each user to show that they are randomly assigned and they do not have any conflicts of subscribing to itself.

- Check if every live user has sent a tweet:
  We check if every live user has tweeted by comparing the tweet table and seeing if every live user has an entry in there.

- Count Retweets:
  We show the amount of retweets to show the functionality works and the randomness.

- Count Mentions:
  We show how many times each user has been mentioned. This is to show the randomness and prove the functionality works.

- Check every tweet has a hashtag:
  We check all the tweets and show that all of them have at least 1 hashtag.

- Count frequency of hashtags:
  We show the frequency of hashtags chosen. They are chosen at random so we have different results every time.

- Logged out users have empty twitter feeds:
  This is to show that the users that were logged out have an empty twitter field. Which is constructed by checking who its subscribed to and building a feed from the original tweet table.

- Logged in users do not have empty twitter feeds:
  This is to show that the users that were logged in have a valid twitter feed, since they all start from blank.

- Check total number of generated tweets:
  Original tweets are generated by each client as per the requirement given (here it is 10).
  We check whether the total number of tweets match the (number of logged in users)*tweets.
  This does not include retweets, since they are not generated but simply repeated.

Rahul Bhatia 5993-0121
Foram Nirmal 6219-9417

- Check if all the tweets are delivered to the live subscribers:
  Here we check the most important part. For each user we query and extract all its subscribers. We then extract all the original tweets generated by the user. Then we use calls and check whether all of the original tweets exist in the feeds of its live subscribers. This ensures the whole twitter engine works.

- Check if tweets with mentions go only to those users:
  We check if the tweets with mentions go only to those users if they are live and not to all subscribers.

- Build the feeds for the users that were logged out:
  We reconstruct the feeds for the logged out users by building them up from the tweets table by comparing who they are subscribed to. We have not printed this result as it prints too many tweets and takes over the screen, however you can uncomment the #Enum.each(feed, fn x → IO.inspect(x) end) line and see the results.

- Delete account:
  We delete a single account and remove all its references from the tweets table. A successful delete passes this test.