DOSP Project 4 Part I

Hrishikesh Balaji (UF ID 3299-8859)

Description:

This project's purpose is to build a Twitter clone and a client tester/simulator. The problem statement is to create an engine that will be linked with Web Sockets to enable complete functionality in part II. The client and server parts will be run in separate Genserver processes. Several client processes are launched and managed by a single server. The following functionalities have been implemented in the twitter server:

- Register an Account.
- Delete an Account.
- Subscribe to a users' tweets.
- Re-tweet, so that subscribers can look at where you got a good tweet from.
- 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)

Implementation Details:

The Twitter engine and clients are two distinct GenServer processes. The client genserver generates many clients. The Server Client requests all functionality from the server. The Server performs the computation and responds to the client. We're utilizing ets tables to simulate database behavior. To help the system, the following tables were created:

- **tweetsMade**: this table will hold all the tweets made by one user. The key would be the user id.
- **following**: holds the user id as the key and the number of users that the given user follows.
- **Followers:** holds the information regarding the users that follow a particular user
- **mentionsHashtags**: holds the hashtag or mention as the key and the tweets containing that hashtag or mention for querying purposes.

The following files are present in out implementation:

1. **main.ex**: This file contains the main simulation component of the project. As command line parameters, this file reads the number of users and the amount of tweets that a user must make. The start function in the main file replicates the distribution of tweets, people following other users, tweeting, and querying. The simulation can be divided into three parts:
    - **Pre-tweeting**: where the users are created and the following tables are populated
    - **Tweeting**: where the every user send as many tweets as mentioned from the input of the command line

- **Post-Tweeting:** where the client is allowed to query the tweets received based on the hashtags, mentions and the tweets subscribed to.

2. **Server.ex:** This file hosts all the functions for the twitter engine implementation responsible for processing the tweets and distributing the tweets. The engine stores all the required information in the ets tables. The engine communicates with the ets tables to retrieve the required data and sends it to the client as and when the request comes in from the client.

3. **Client.ex:** this file hosts the call backs for initiating the tweeting and other functionalities for a single client. The tweeting is always initiated from the client and response are given in the processing from the server.

How to run:

mix proj4.exs <Num of clients> <no of tweets>

Num of clients: Users to be simulated

Num of tweets: Tweets sent by every user

Test cases for the functionalities have been implemented and are placed in the proj4_test.exs file. Run the following command to run the test cases:

- mix test

To run individual test cases, run:

- mix test –only <test case tag> . For example, to run the first test case , type in the following command: mix test –only testCase:1

What is being printed:

We print the following the following values in the output screen:

- User creation confirmation
- User tweeting confirmation
- Messages received through the live feed are displayed directly in the feed
- User re tweeting confirmation
- User deletion confirmation

Performance matrix(milliseconds):

The performance of the system is greatly dependent on the number of users and the number of tweets that a user must send across. As displayed in the graph, the total time required to simulate the tweeting process increases greatly with the number of users in the system. Since, the tweeting process is asynchronous, we are confident that the system would be able handle many more users, but with increase time requirement.