**DOS Project 4 Part I**

Milind Jayan (UF ID 8168 9617)

Sagar Jayanth Shetty (UF ID 4351 7929)

**Brief Description**:

The goal of this project is to implement a Twitter Clone and a client tester/simulator. The problem statement is to implement an engine that (in part II) will be paired up with Web Sockets to provide full functionality. The client part and the server part are to be simulated on separate Genserver processes. Several client processes are spawned and are handled using a single server.

The following functionalities have been implemented in the twitter server:

* Register Account
* Delete Account
* Subscribe to users’ tweets
* Re-tweets so that subscribers get an interesting tweet you got by other means
* 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 the clients are separate GenServer processes. Multiple clients are spawned from the client genserver. The Server Client calls the server for all the functionality. The Server does the computation and provides a response to the client. To emulate the behavior of databases, we are using ets tables. The following tables were created to support the system:

* tweetsMade – this table will hold all the tweets made by one particular 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 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 hosts the main simulation part of the project. This file reads in the number of users and the number of tweets that a user has to make as command line arguments. The start function in the main file simulates the distribution of the tweets, users following other users, tweeting and querying as well. The whole simulation can be broken down into three parts, which are,

* 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.

1. 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.
2. 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 has to 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number of Clients | Number of tweets | Time till tweeting starts(create users, follow users etc.) | Time to tweet | Time to query | Total Time |
| 5 | 5 | 0 | 10 | 0 | 10 |
| 10 | 5 | 34 | 20 | 0 | 54 |
| 50 | 5 | 166 | 62 | 10 | 238 |
| 100 | 5 | 478 | 510 | 16 | 1004 |
| 500 | 5 | 523 | 3539 | 250 | 4312 |
| 1000 | 5 | 732 | 15279 | 906 | 16917 |
| 1500 | 5 | 1893 | 107468 | 1954 | 111315 |
| 2000 | 5 | 3214 | 254171 | 3313 | 260698 |