DOS Project Report
Twitter Simulator

Team Members :

Desikan Sundararajan - 5615-9991
Madhura Basavaraju - 6794-1287

How to Run

Running the simulation :
mix run proj4.exs numNodes numRequests

Running the test cases :
mix test

Implementation

We have divided our simulator into three components - the Twitter Engine, the ClientSupervisor and the Client worker Genserver. We have used Erlang's in-memory database storage ETS, to store and retrieve tweets from memory so that they can be shared across users.

The Twitter Engine is responsible for distributing tweets to users, and for storing tweets of users. The Client Supervisor sends requests to the engine on behalf of each user(client) worker GenServer. It maintains a list of all the users and carries our requests and gets responses from the engine. All of the functionalities that involve querying tweets, retweeting, adding followers etc are carried out by the engine.

Our schema works by having three main tables.

1) :registrations (username=>password):
   We use it for maintaining the list of registered users.

2) :users (user=>[message,hashtags,mentions,time_stamp,retweet_ctr,msg_owner]) :
   This mapping is for storing the list of relevant tweets for a particular user. Each tweet is a tuple consists of a number of parameters, such as the number of times it has been retweeted, the hashtags, mentions etc.

3) :userfollowing (user=>[user_following]):
   This mapping is for storing the list of users a particular user is following.

Functionalities

The list of functionalities that have been implemented are as follows :

1) Send Tweet : Posting tweet for all followers
2) Receive Tweet : Getting tweets from all subscribers
3) Searching Tweets Subscribed to : Querying these tweets
4) Search by Hashtag : This feature is global. A user can search tweets even if he's not subscribed to it. (just like it works in Twitter)
5) Search my mentions : Searching all tweets a user has been mentioned in. Again a global feature.
6) Store Tweet : Storing tweets in memory
7) Add Followers : Following interested users
8) Retweet messages : Repost a message that a user is interested in
9) Login status : A user is logged in by validating credentials against that stored in the registration table
10) User Registrations : Username and password stored in ETS memory
11) Live Tweeting : Sending out live notifications to a user's followers if they are already logged in.

The correctness can be verified using test cases that have been written for the same.

Working

The features implemented are all functions that the Twitter Engine handles. These have been handled in test cases where we call the Engine to verify correct results. Apart from this, we have implemented a simulation, where given the number of users and the number of tweets, each user sends out that many messages concurrently to his followers. This was implemented by assigning the usernames, follows and messages randomly. The message generated is of the form "Tweet (number) from user (username)" and they are sent out simultaneously using asynchronous tasks.

Performance analysis

Any number of users may be created. All users are initially given the default status of being online. The live subscribers of the user posting a tweet receive the tweet immediately in the form of a notification. This notification is saved in the user's state so as to prohibit any kind of querying required from the database in order to retrieve the message.

We were able to handle upto 1000 requests and 1000 users.
A total of 1000 users can make 1000 tweets with all the users being live in 9562 milliseconds.

500 users can post 500 tweets simultaneously (all users live) in 6360 milliseconds.