COP5615 - DOSP Project 4 - Part 1

Team Members

Akash Pinnaka - UFID 1009-4703
Thilak Reddy Kanala - UFID 8800-3203

Architecture

We utilize genservers to instantiate the users and the twitter engine. The twitter engine is a single genserver while each user is a separate new genserver. The communication between these entities takes place over both asynchronous and synchronous calls (“cast” and “call”) respectively. To register an account, the user calls the engine with it’s data and the engine stores the user data in its own loopdata, the engine then updates all the nodes which makes every user aware of the new registration. We use node and user synonymously in this report.

For the purpose of testing, each user is named 1, 2, 3 .. etc. During simulation we generate users with a random amount of subscriptions. We also generate random Tweets for the purpose of simulation where each tweet is of the format: TweetConent = “Hello from User <x>! <#hashtag1> <#hashtag2> .. etc <@1> <@2> … etc” where each TweetContent consists of a random number of mentions and hashtags between 1-10.

While we simulate the engine, we also let each user tweet a random number of tweets which also includes retweets. The retweet mechanism works by selecting a tweet from a user’s own live feed and posting that with the type changed from “ORIGINAL” to “RETWEET”

The following are the different custom data structures utilized:

- ➔ UserData: {Name, NodePID, List of Users Subscribed To, List of Received Tweets}
- ➔ TweetData: {Author, Content, Type (“ORIGINAL” / “RETWEET”), List of Hashtags, List of Mentions}
- ➔ ZipfData: {Name, Subscriber Count, User Rank, Number of Tweets Posted by User}

Demonstrating Functionality

1. Registering a user

    a. We register a user by calling a callback function of the engine, which in turn stores the user data in it's loopdata

    b. The user data is of the format UserData = {Name, NodePID, List of Users Subscribed To, List of Received Tweets}

```
register_user(UserData) ->
    {_, NodePID} = gen_server:start_link(?MODULE, [], []),

    {Name, _NodePID, Subscribed, Tweets} = UserData,
    NewUserData = {Name, {nodePID, NodePID}, Subscribed, Tweets},

    gen_server:call('Engine', {register_user, NewUserData}).
```

Fig 1. Register User function

2. Sending a Tweet and **subscribing** to other user's tweets

    a. start the engine
    b. register 5 Users, making sure each user is subscribed to a few random users
    c. generate and tweet a random tweet from User 1
    d. observe the data in the engine loopdata

The result is as expected, we observe 5 registered users each with a random set of subscriptions, and one tweet from User 1.

```
14    start(NNodes) ->
15        engine:start(),
16        timer:sleep(500),
17
18        _AllUsers = register_all_users(NNodes, NNodes, []),
19        timer:sleep(500),
20
21        Tweet1 = get_random_tweet(1, NNodes),
22        engine:send_tweet_from_Nth_Node(1, Tweet1),
23
24        engine:show_loopdata(),
25        timer:sleep(500).
```

Fig 2. Code to demonstrate Tweet and Subscribe

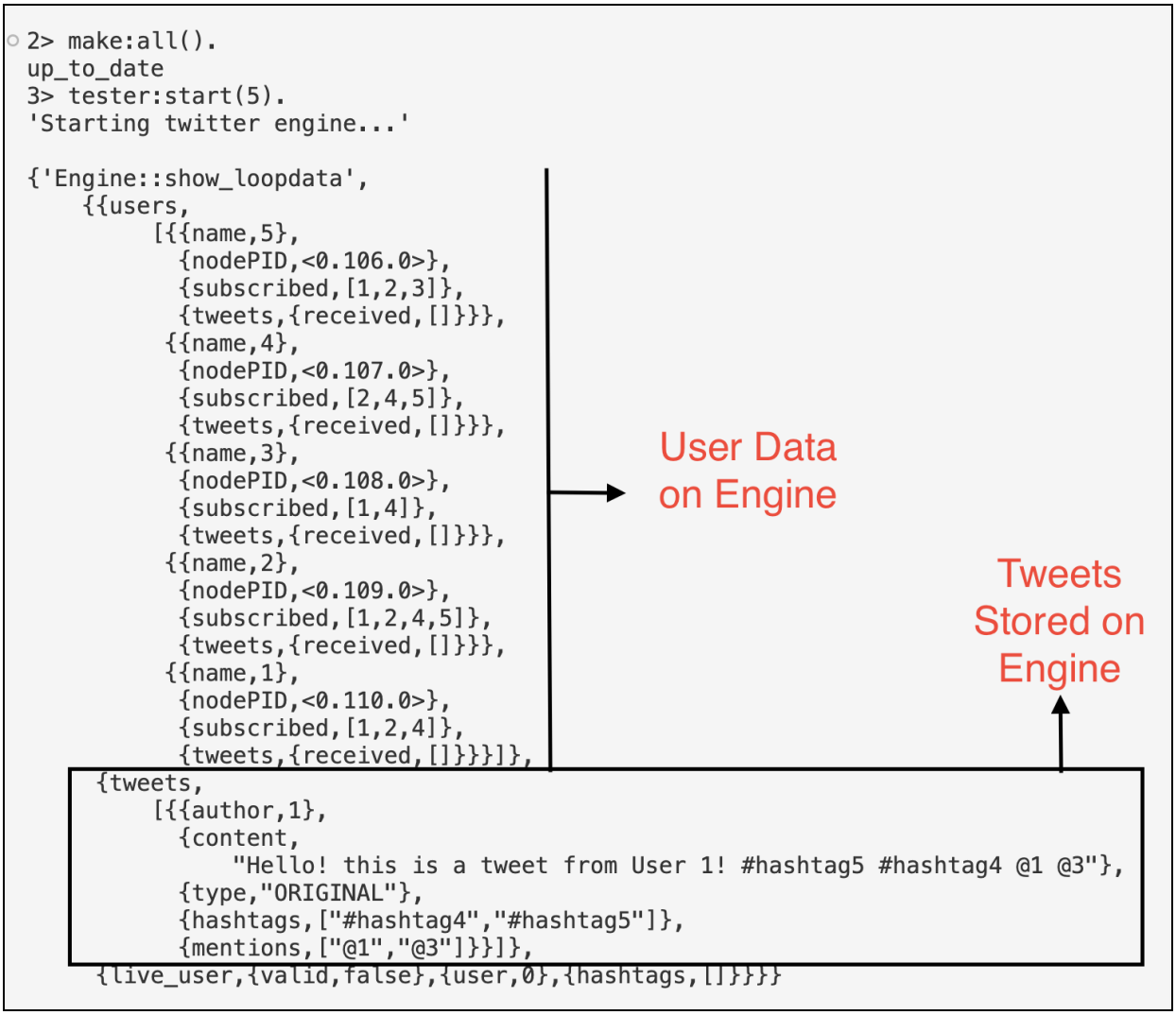

Fig 3. Result of Tweet and Subscribe Demonstration

3. Posting **Retweets** (Tweets acquired from other sources can be retweeted)
    a. start the engine
    b. register 5 users
    c. post a random tweet from User 1 mentioning User 3
    d. query mentioned tweets to update User 3's live feed
    e. then retweet the tweet received by User 3 from User 1
    f. observe the result on the engine loopdata

The result is as expected, User 3 receives the tweet from User 1 and retweets which can be observed in the engine's loopdata

```
14  start(NNodes) ->
15      engine:start(),
16      timer:sleep(500),
17
18      _AllUsers = register_all_users(NNodes, NNodes, []),
19      timer:sleep(500),
20
21      Tweet1 = get_random_tweet(1, NNodes),
22      engine:send_tweet_from_Nth_Node(1, Tweet1),
23      timer:sleep(500),
24
25      node:query_mention_tweets(3),
26      timer:sleep(500),
27
28      engine:send_random_retweet_from_Nth_Node(3),
29      timer:sleep(500),
30
31      engine:show_loopdata(),
32      timer:sleep(500).
```

Fig 4. Code to demonstrate retweets

Fig 5. Result of demonstrating retweets

4. Delivering tweets **live** to the user's feed when user is connected
   Demonstrates the **Live Feed** functionality along with functionality for querying tweets **subscribed** to, tweets with **specific hashtags**, tweets in which the user is **mentioned**

   a. We start the engine
   b. Register 5 users to the network
   c. Register User 2 as a live user who is interested in "#hashtag5" and subscribed to User 1
   d. We then generate and tweet 3 random tweets from User 1, User 3 and User 4 ensuring that User 2 is involved in some way (either subscription, hashtag matching or mentioned)
   e. We observe the live feed as the result

The result is as expected and User 2 receives the tweet from User 1 due to subscription, receives the tweet from User 3 because "#hashtag5" matched and User 2 is mentioned, and finally receives a tweet from User 4 because User 2 in mentioned.

```
14  start(NNodes) ->
15      engine:start(),
16      timer:sleep(500),
17
18      _AllUsers = register_all_users(NNodes, NNodes, []),
19      timer:sleep(500),
20
21      LiveUser = {live_user, {valid, true}, {user, 2}, {hashtags, ["#hashtag5"]}},
22      engine:register_live_user(LiveUser),
23
24      Tweet1 = get_random_tweet(1, NNodes),
25      engine:send_tweet_from_Nth_Node(1, Tweet1),
26
27      Tweet2 = get_random_tweet(3, NNodes),
28      engine:send_tweet_from_Nth_Node(3, Tweet2),
29
30      Tweet3 = get_random_tweet(4, NNodes),
31      engine:send_tweet_from_Nth_Node(4, Tweet3).
32
```

Fig 6. Code to demonstrate live feed and queries

```
3> make:all().
up_to_date
4> tester:start(5).
'Starting twitter engine...'

{live_user_registered,{live_user,{valid,true},
                                 {user,2},
                                 {hashtags,["#hashtag5"]}}}

{'livefeed::subscribed_tweet',
    {{author,1},
     {content,
         "Hello! this is a tweet from User 1! #hashtag2 #hashtag1 @4 @1 @3"},
     {type,"ORIGINAL"}}}

{'livefeed::hashtag_matched_tweet',
    {{author,3},
     {content,
         "Hello! this is a tweet from User 3! #hashtag1 #hashtag4 #hashtag5 #hashtag3 @2 @4 @1 @3 @5"},
     {type,"ORIGINAL"}}}

{'livefeed::mentioned_tweet',
    {{author,3},
     {content,
         "Hello! this is a tweet from User 3! #hashtag1 #hashtag4 #hashtag5 #hashtag3 @2 @4 @1 @3 @5"},
     {type,"ORIGINAL"}}}

normalReply
{'livefeed::mentioned_tweet',
    {{author,4},
     {content,
         "Hello! this is a tweet from User 4! #hashtag1 #hashtag2 #hashtag4 #hashtag3 @5 @4 @2 @1"},
     {type,"ORIGINAL"}}}
```

Fig 7. Result of demonstrating live feed and queries

Simulation and Performance Report

1. Simulating users with periods of live connection and disconnection

   ➔ We were able to simulate a maximum of 30,000 users in under 5 seconds CPU Time with each user posting between 1-50 tweets.

   ➔ We were able to simulate maximum of 80,000 users in under 5 seconds of CPU Time for only registering the users into the system

   ➔ By allowing live feed for a certain amount of time for few random users, we were able to simulate 10,000 users in under 5 seconds CPU Time with each user staying receiving live feed for a random amount of milliseconds between 50-100.

2. Simulating a Zipf Distribution

   ➔ The Zipf distribution simulation is achieved by by generation a distribution with the values of
      - The total number of subscribers for each user
      - The total number of tweets in that session
      - The Zipf Constant

   ➔ We assign a rank to each user based on the number subscribers for that user. A lower rank indicates a higher number of subscribers.

   ➔ Tweets Posted = $\frac{(Total\ Tweets\ in\ Session) * (Zipf\ Constant)}{Rank}$

   ➔ The Zipf constant is usually empirically derived from a distribution, but for simulating the distribution we assigned to a few commonly observed values

3. Zipf Distribution Graphs

➔ Graphs with 1000 Users, 10000 Tweets in Session and Zipf Constant = 0.1

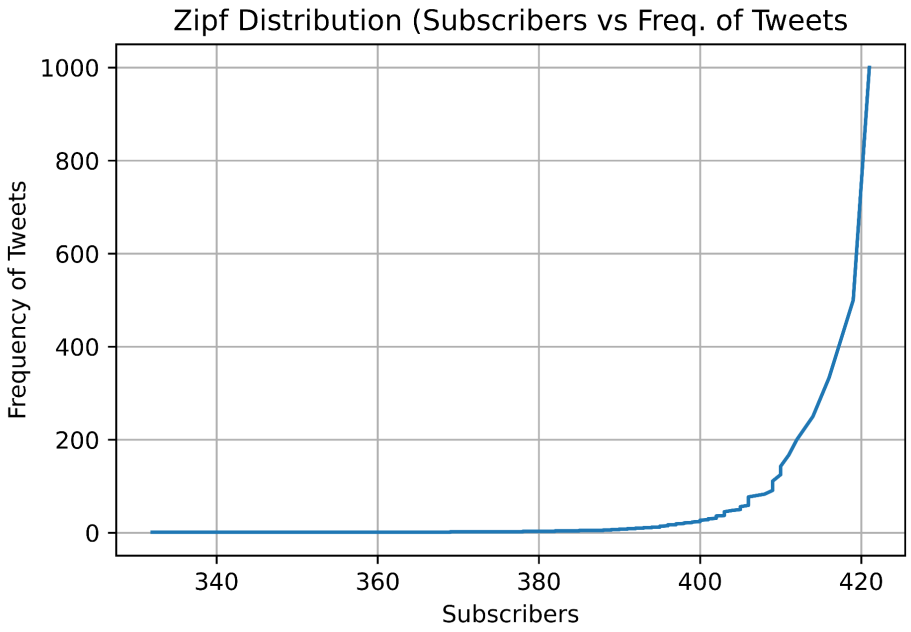

Fig 9. Graph of Subscribers vs Freq. of Tweets (N=1000, ZipfC = 0.1)

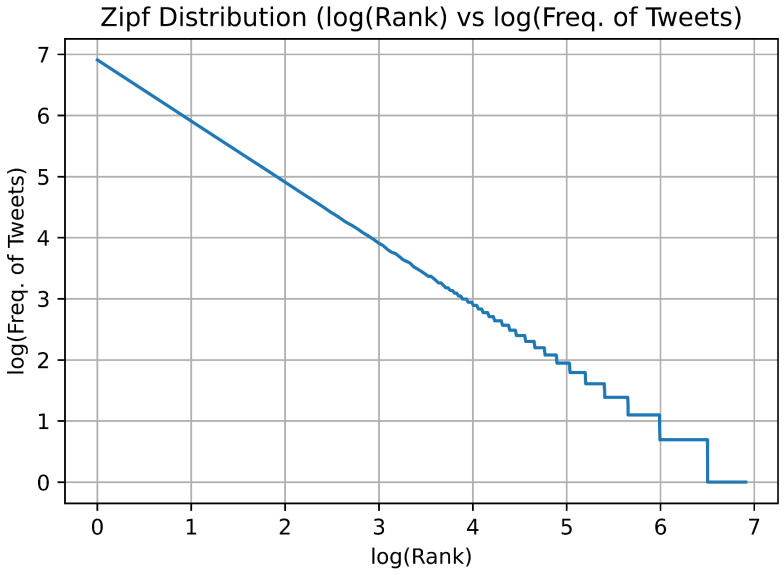

Fig 10. Graph of log(rank) vs log(Freq. of Tweets) (N=1000, ZipfC = 0.1)

➔ Graphs with 1000 Users, 1000 Tweets in session and Zipf constat = 0.5

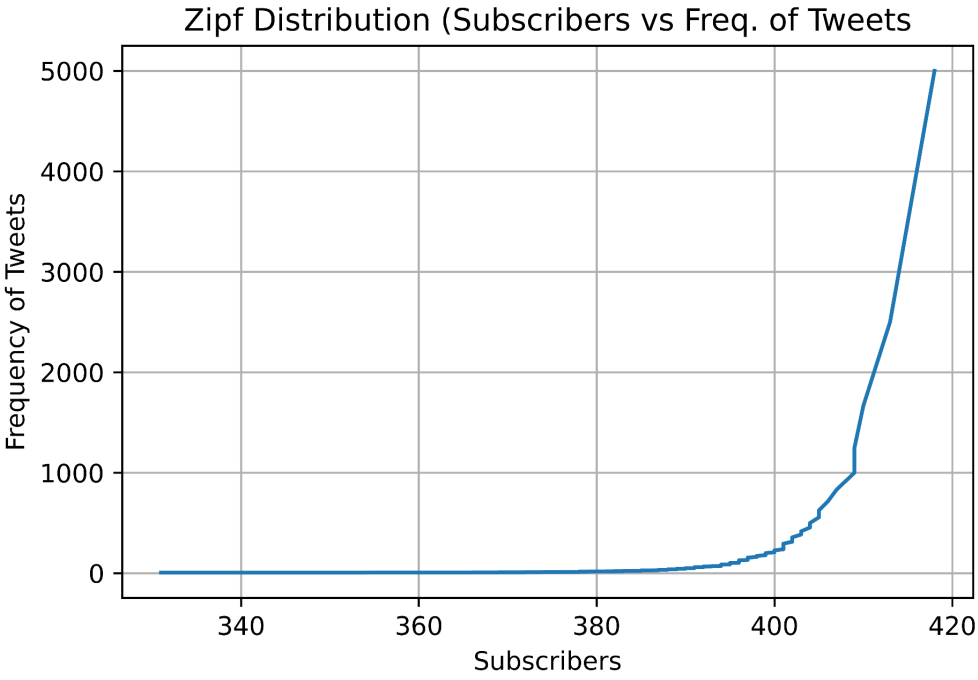

Fig 11. Graph of Subscribers vs Freq. of Tweets (N=1000, ZipfC = 0.5)

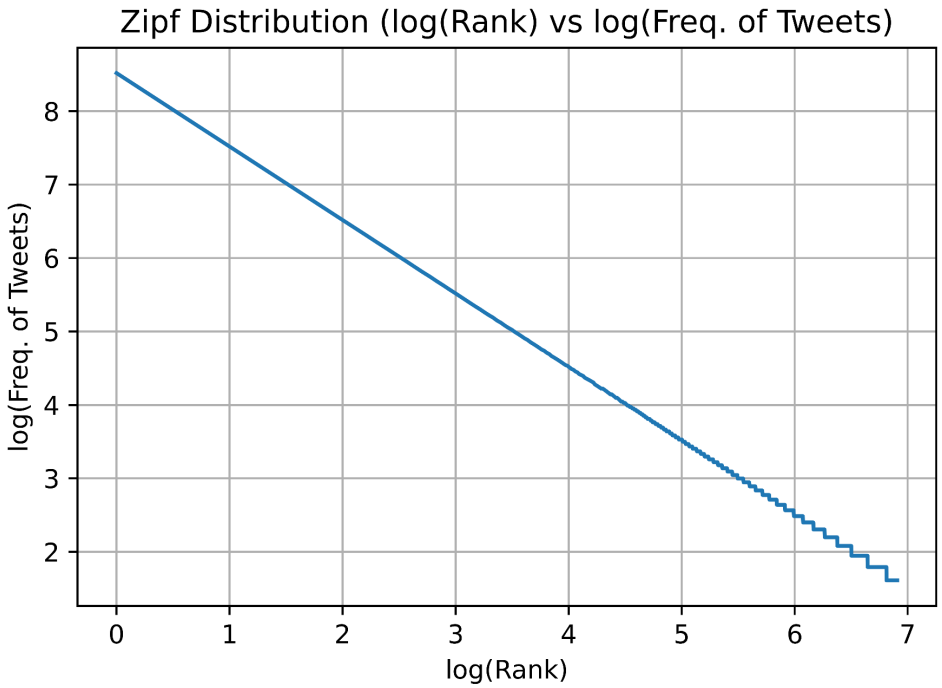

Fig 10. Graph of log(rank) vs log(Freq. of Tweets) (N=1000, ZipfC = 0.5)