Distributed Systems - Project 4 Part 1 Twitter Simulator

Geetanjli Chugh(21885647) | Yash Sinha (15618171)

1. Modules

The project consists of 5 modules:

1. Main module
2. Server module
3. ETS module
4. User module
5. Test module

1.1. Main Module

The main module handles the initialization of the server process or the client process depending on the user arguments. Two forms of arguments are allowed:

1. ./twittersimulator 1
   This starts the server process
2. ./twittersimulator numusers serverip
   This starts the client process. Eg. ./twittersimulator 5 10.20.78.254

1.2. Server Module

The server module handles the data get and put API calls from the clients. It gets the requests from the clients, processes the requests from ETS tables and sends back the results to the client

1.3. ETS Module

This handles the ETS tables methods. It is responsible for inserting, fetching and deleting the data from the ETS Table.

1.4. User Module

This is the client module. It maintains the properties of the user and handles the functions to take random actions like tweet, retweet, follow and queries.

1.5. Test Module

This is the test module which creates the initial users, their followers' lists, creates workers for the active users and starts the simulation.

2. ETS tables

There are 3 ETS tables used in the project. User table, Tweet table and the Hashtag table.

The user table maintains the user data. The user table contains the following fields:

userid

name
password
followers – list of followers
following – list of followings
tweets – list of tweetids which the user tweeted/retweeted
mentions_tweets – list of tweetids where the user was mentioned

The tweet table contains the tweet data. The fields are:
tweetid
timestamp – time at which the tweet was created
tweet – tweet content
userid – id of the user who created the tweet
original_opid_tweetid – if this a retweet this contains original userid and tweetid
retweets_list – list of tweet ids which retweeted this tweet

The hashtags table contains the tweets corresponding to respective hashtags. The fields are:
hashtag
tweeted_list – list of tweetids which contained this hashtag

3. Application Flow

3.1. Initially, we boot up the server process using the commands mentioned in section 1.1. The server module creates the initial tables – user, tweet, hashtags.

3.2. When we start the client process, it does a Node.connect on the server IP. It then spawns the test module process. The test module process creates the number of users given as the argument. The create user requests are sent to the server. When the server creates the users, followers list of each of the users are created based on the Zipf distribution. Zipf distribution implies that the user with the smallest userid has the largest number of followers. Correspondingly the following lists of the users are populated.

3.3. Once the followers list and followings list is populated, the test module starts the simulation. The simulation is designed in such a way that at every simulation point, 50% of the total users are active. One actor is created for each of the active user. To maintain the fact that users with more followers will have more tweets, we calculate the action interval of each of the user as follows:
*interval = round(100 + (userid * 100))*
where 100 and 200 are hyperparameters.
This implies that a user with a smaller userid will have to wait less to take a random action.

3.4. The user process then starts doing random actions. The random actions include: *Tweet, Retweet, Follow and Query.* Three types of queries are allowed: tweets subscribed to, tweets with specific hashtags, tweets in which the user is mentioned.

Tweet:

When the user creates a tweet, we create a random tweet description. We select some random hashtag and some random users and append to a randomly generated string. Hashtagging and Mentioning other users are optional in the tweet description i.e. there can be a tweet which doesn't contain a hashtag or mentioned a user. Once the description is generated, we send a request to the server to add the tweet to the tweet ETS table. The server auto-increments the tweetid and adds the entry. We also append this tweetid to list of user tweets.

Retweet:

To retweet, we first select a random tweet which was not created by this user. We then extract the description and the original tweet id and original poster's id. We create a new tweet object, increment the tweetids at server and add the entry to the tweet ETS table. We also append this tweetid to list of user tweets.

Follow:

For follow, we select a random user and follow it. The follower's list of the random user is updated and the following list of the follower is updated. This is the same as subscribing to user tweets.

Query:

Hashtags: We extract all the tweetids of the queried hashtag and then extract tweet description and other required information from the tweet ETS table and send back the result to the user

Tweets subscribed to: We extract the following list of the user and then fetch their tweets from the tweet ETS table and send it back to the user.

Tweets mentioned in: We extract the mention list of the user which contains the tweetids where the user was mentioned and get the tweet descriptions and send it back to the user.

3.5. We maintain the live connections and disconnections using the test module simulation. Every 20 seconds, we take random users from the userlist and make them live. When the user comes live, we fetch its feed from the server. The feed contains the tweets that the user sent, the following tweets and the tweets in which the user was mentioned. If the user is live and is mentioned in some other user's tweet or someone he is following creates a new tweet/retweet, we update the feed of the live user so that he can get the updated tweets.

3.6. We have designed the application in such a way that the random action of tweeting is given the most weightage followed by retweet, query and follow. The zipf distribution mentioned in section 3.3. helps us control which user takes most frequent actions. For

eg. A user with a userid 1 will wait for 200 ms before taking a new random action. A userid 5 will have to wait 600 ms before doing so.

4. Performance measures

We have analysed the following measures to get the performance of the application.
Parameters:
Percentage of users which are active: 50%
Simulation interval: 20 seconds
Action frequency: *round(100 + (userid * 100))*

Metrics:

1. Tweet rate
   Tweet rate is defined as the number of tweets created per second. When the number of users is small, the tweet rate will be less.

2. Random-action rate
   Since tweeting and retweeting are not the only actions which load up the server, we also analyzed the number of random actions per second.

Number of users	Tweet Rate	Random-action Rate
100	9	14
1000	10	15
5000	5	9
10000	4	8
20000	3	5

100 users:

```
Time elapsed: 10277, Num users: 100, Num tweets: 96, Tweet rate: 9 per second, Num random actions/second = 14
Time elapsed: 10764, Num users: 100, Num tweets: 99, Tweet rate: 9 per second, Num random actions/second = 14
Time elapsed: 10965, Num users: 100, Num tweets: 99, Tweet rate: 9 per second, Num random actions/second = 14
```

1000 users:

```
Time elapsed: 21577, Num users: 1000, Num tweets: 205, Tweet rate: 10 per second, Num random actions/second = 15
Time elapsed: 25238, Num users: 1000, Num tweets: 243, Tweet rate: 10 per second, Num random actions/second = 15
Time elapsed: 26897, Num users: 1000, Num tweets: 259, Tweet rate: 10 per second, Num random actions/second = 15
Time elapsed: 27781, Num users: 1000, Num tweets: 265, Tweet rate: 10 per second, Num random actions/second = 15
```

5000 users:

```
Time elapsed: 10406, Num users: 5000, Num tweets: 48, Tweet rate: 5 per second, Num random actions/second = 9
Time elapsed: 10713, Num users: 5000, Num tweets: 50, Tweet rate: 5 per second, Num random actions/second = 9
Time elapsed: 10914, Num users: 5000, Num tweets: 50, Tweet rate: 5 per second, Num random actions/second = 9
Time elapsed: 11732, Num users: 5000, Num tweets: 53, Tweet rate: 5 per second, Num random actions/second = 9
```

10000 users:

```
Time elapsed: 5582, Num users: 10000, Num tweets: 24, Tweet rate: 4 per second, Num random actions/second = 8
Time elapsed: 5822, Num users: 10000, Num tweets: 24, Tweet rate: 4 per second, Num random actions/second = 8
Time elapsed: 6207, Num users: 10000, Num tweets: 24, Tweet rate: 4 per second, Num random actions/second = 7
```

20000 users:

```
Time elapsed: 12264, Num users: 20000, Num tweets: 38, Tweet rate: 3 per second, Num random actions/second = 6
Time elapsed: 13313, Num users: 20000, Num tweets: 40, Tweet rate: 3 per second, Num random actions/second = 5
Time elapsed: 13706, Num users: 20000, Num tweets: 41, Tweet rate: 3 per second, Num random actions/second = 5
Time elapsed: 13906, Num users: 20000, Num tweets: 41, Tweet rate: 3 per second, Num random actions/second = 5
```

3. Network size
   The maximum network size we tried was 500000 users. One of the bottleneck of the program is creating initial follower list to maintain the zipf constraint. For each of the 500000 users the application has to create the required number of followers which slows up the application drastically. When we ignored the *followers* constraint, the application was able to generate tweets for 2 million users too.

4. CPU Utilization
   The application was run on a 8-core machine. The CPU utilization was around 700%.

5. Failure handling
   In case the server crashes, the client stops sending the tweets.

5. Running the application

1. Extraction the zip folder twittersimulator.zip
2. cd to the **twittersimulator** folder
3. Run ***mix escript.build***
4. Starting the server:
   Run: ***./twittersimulator 1***
   This will return the ip of the server too. Let this be **serverip**
5. Starting the client
   Run: ***./twittersimulator numusers serverip***
   Eg. ./twittersimulator 100 10.20.78.254

6. Interpreting the outputs

The server and client processes display different outputs. The server displays the performance metrics in certain interval of time.

```
Yashs-MacBook-Pro:twittersimulator working yash$ ./twittersimulator 1
Serverip: 10.20.78.254
```

```
Time elapsed: 1919, Num users: 1000, Num tweets: 11, Tweet rate: 6 per second, Num random actions/second = 9
Time elapsed: 2434, Num users: 1000, Num tweets: 16, Tweet rate: 7 per second, Num random actions/second = 11
Time elapsed: 4834, Num users: 1000, Num tweets: 46, Tweet rate: 10 per second, Num random actions/second = 15
Time elapsed: 5521, Num users: 1000, Num tweets: 53, Tweet rate: 10 per second, Num random actions/second = 16
Time elapsed: 7019, Num users: 1000, Num tweets: 73, Tweet rate: 11 per second, Num random actions/second = 17
Time elapsed: 8438, Num users: 1000, Num tweets: 95, Tweet rate: 11 per second, Num random actions/second = 18
Time elapsed: 12398, Num users: 1000, Num tweets: 157, Tweet rate: 13 per second, Num random actions/second = 20
```

The client displays the feed of the user. This is feed is updated for the user as mentioned in section 3.5.

```
Yashs-MacBook-Pro:twittersimulator working yash$ ./twittersimulator 1000 10.20.78.254
7.485470860550341
Getting user list
User: user5 update: created tweet ------------------------------
My tweets (user5):
2017-12-03 01:31:20.925590Z ||| I  tweeting!  am  Here  @user2 @user1 @user11 @user29 @user3 #dos  #uf  #twitter ||| Number of retweets: 0
2017-12-03 01:31:21.525107Z ||| tweeting!  Here  am  I  @user35 @user41 ||| Number of retweets: 0
2017-12-03 01:31:22.127491Z ||| I  tweeting!  Here  am  @user25 @user37 @user35 @user23 @user15 @user40 @user31 @user4 @user18 @user30  #uf #twitte
r #dos ||| Number of retweets: 0
2017-12-03 01:31:22.726030Z ||| tweeting!  I  Here  am  @user38 @user32 #twitter  #dos ||| Number of retweets: 0 retweet op: user1
-----
My mentions(user5):
2017-12-03 01:31:21.121137Z ||| tweeting!  am  Here  I  @user5 @user25 @user49 ||| op: user1
2017-12-03 01:31:21.221613Z ||| Here  I  am  tweeting!  @user26 @user5 @user14 @user15 @user10 @user35 @user31 @user22 @user11   #dos ||| op: user4
2017-12-03 01:31:22.724217Z ||| Here  am  tweeting!  I  @user25 @user47 @user21 @user15 @user48 @user3 @user35 @user46 @user44 @user5  #uf ||| op:
user9
----
My followings(user5):
2017-12-03 01:31:20.127812Z ||| tweeting!  I  Here  am  @user28 @user37 @user18 @user35 @user24 @user32 @user38 #dos ||| op: user1
2017-12-03 01:31:20.521926Z ||| I  am  tweeting!  Here  @user43 @user35 @user2 @user14 @user36  #uf  #dos #twitter ||| op: user1
2017-12-03 01:31:20.722168Z ||| tweeting!  I  Here  am  @user38 @user32 #twitter  #dos ||| op: user4
2017-12-03 01:31:20.922043Z ||| I  Here  am  tweeting!  @user49 @user9 @user6 @user41 @user11 @user32 @user45 #twitter  #dos ||| op: user1
2017-12-03 01:31:21.121137Z ||| tweeting!  am  Here  I  @user5 @user25 @user49 ||| op: user1
2017-12-03 01:31:21.823849Z ||| Here  am  I  tweeting!  @user19 @user40 ||| op: user20
2017-12-03 01:31:21.922243Z ||| I  Here  am  tweeting!  @user28 @user31 #uf ||| op: user1
2017-12-03 01:31:22.122522Z ||| tweeting!  I  am  Here  @user32 @user10 @user28 @user15  #dos  #uf #twitter ||| op: user1
2017-12-03 01:31:22.523052Z ||| tweeting!  Here  am  I  @user22 @user31 @user43 @user42  #twitter ||| op: user1
```