README

Team Members

- Arijit Dutta (UFID:55889097)
- Meghamala Gupta (UFID:65270126)

Steps to Run

dotnet add package Akka –version 1.4.25
dotnet add package Akka.FSharp –version 1.4.25
dotnet fsi twitterSimServer.fsx <serverIP>
dotnet fsi twitterSimClient.fsx <serverIP> <noOfUsers>

- serverIP - IP of the machine where the "twitterSimServer.fsx" is running.
- noOfUsers - An integer which specifies the number of users interacting with the twitter server.

Introduction

Our architecture uses Actor Model in F# to implement the twitter simulator- twitter backend engine and a client/tester simulator.

- We have implemented the registration functionality for the users. As soon as a user is registered, their status is set to online.
- The status of an user is changed to online and offline periodically, every 5 seconds.
- Users can subscribe to each other according to the Zipf distribution. This means that users with rank 1 will have (n-1) subscribers where n = no of users. User with rank 2 will have (n-1)/2 subscribers and hence forth. The rank to each user is assigned randomly after all users are created.
- Every user, if online, can tweet or retweet. The total number of tweets(including retweets) which a user can make per cycle is dependent on its subscribers count and hence follows Zipf distribution.
- For testing purpose, a user can tweet every 1 second. A tweet can be any of the following:
  - Without any hashtags or mentions.
  - Retweet a random tweet from it's feed.
  - Have a hashtag chosen randomly from a given list of hashtags.
  - Have a mention chosen randomly from the list of registered users.
- As soon as an user tweets something, their tweet gets updated to all of their subscriber's feed.

- For every tweet, the tweet is immediately parsed to extract the hashtags and mentions(if any) present in it. If there is any hashtag present, a record of the tweetId where it was mentioned is added in a hashmap. Same happens for any mentions present. The tweet also appears in the mentioned user's feed.
- When the tweets are shown in the user's feed, the user also has the option to retweet some of the random tweets from this list.A retweet is not parsed again but is treated as fresh tweet from the user. This means that the retweet counts towards the total no of tweets and is added to the feeds of all the user's subscribers.
- An user also has the ability to search for any hashtags. The tweets which contain the searched hashtag are shown on the user's feed. In case the hashtag is not used in any tweet yet, no tweets are shown.
- An user also has the ability to search for his mentions(tweets where he has been mentioned). All the tweets where he was mentioned is shown on the user's feed. In case he has not been mentioned in any tweets yet, then we display a message saying " You have not been mentioned yet".
- In the client code, for testing purposes, the query mention or query hashtag requests are generated every 500ms.

Observations

- With the increase in the number of users the average time per registration and follow request decreases.
- With the increase in the number of users the average time per tweet, retweet, query hashtag and query mention increases. This is justified due to the increase in the number of requests per second as the number of users increases.
- We also find that the average time per query hashtag and per query mention is almost the same given a particular number of users. This shows the similar implementation of both these functionalities.
- All the above experiments have been performed keeping the number of requests served by the server constant, at 500,000.

Largest Network

We have tested Chord with 7000 nodes and 500k requests. The client code keeps requesting the server for services randomnly unless 500k requests are reached.

Graphs and Outputs

- The output shows the performance statistics which has the total number of requests received by the server and the total uptime. The output also shows the type of the requested service(tweet,retweet,query mention etc), the number of such requests received and the average time in milliseconds per request for each service.
- When the Server is run, after successful execution, the output will be stored by default in a file called "Stats.txt". The maximum requests till which the Twitter Server(twitterSimServer.fsx) will keep listening is kept at 100,000 by default. If therer's a need to change that please edit the 'maxRequests' variable in twitterSimServer.fsx.
- A file called "PerfStats.txt" is present in the folder which has the Performance Testing logs and gives us an overall idea of how the performance changes with the number of users.

Table 1

This table shows the number of requests of each type corresponding to the number of users present in the leftmost column. Total number of requests received is 500,000.

Number of Users	Register	Follow	Tweet	Retweet	Query Hashtag	Query Mention
1000	1000	7054	72745	18926	200548	199751
2000	2000	15499	76833	2628	201558	201513
3000	3000	24465	129134	4130	169550	169722
4000	4000	33782	178154	5725	139362	138979
5000	5000	43357	163267	6119	141261	140997
6000	6000	53102	163977	10776	133545	133164
7000	7000	63040	167727	10667	125650	126116

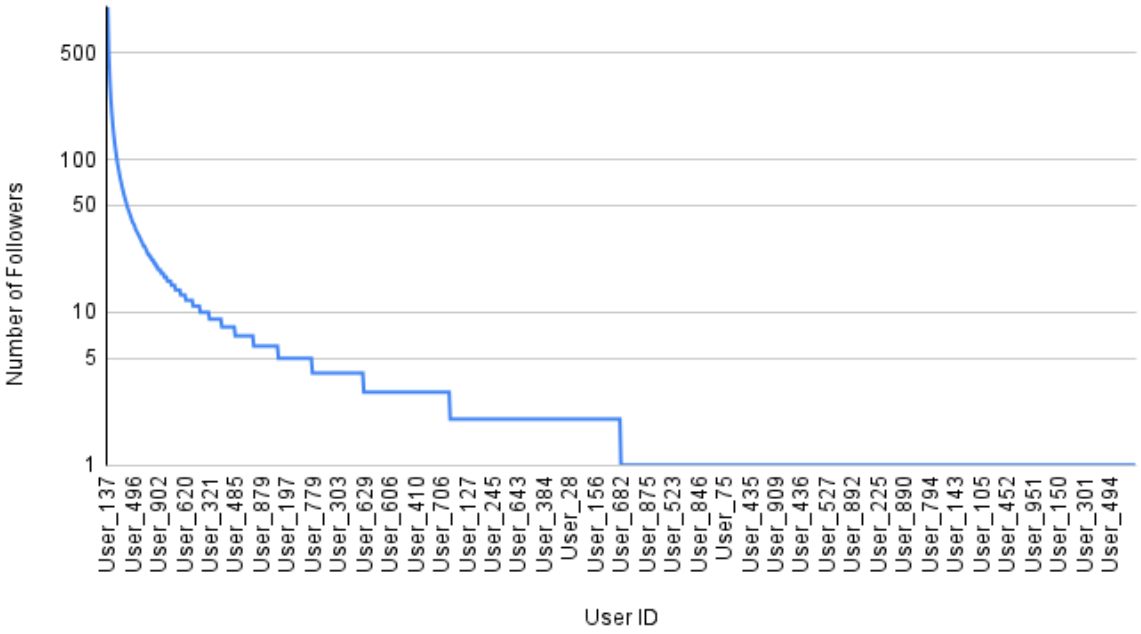
Table 2

This table shows the average time(ms) per request of each type corresponding to the number of users present in the leftmost column. Total number of requests received is 500,000.

Number of Users	Register	Follow	Tweet	Retweet	Query Hashtag	Query Mention
1000	0.890728	0.167077	1.063535	7.083154	0.668541	0.671235
2000	0.575247	0.137031	1.447168	3.092295	0.925999	0.926251
3000	0.380234	0.115693	2.157732	5.184633	2.141945	2.139817
4000	0.344445	0.093192	3.460492	7.609265	4.443098	4.455345
5000	0.275411	0.082679	4.777381	6.840682	5.521287	5.531624
6000	0.222122	0.072009	23.047983	9.363834	9.073171	9.099101
7000	0.212057	0.076572	21.871561	341.310757	11.158314	11.117233

Graph 1

The below graph shows the Zipf distribution of the number of followers of 1000 users. The users are arranged according to their rank in the X-axis.



Sample Output 1

The below image shows a screenshot from the "Stats.txt file". This file will keep the output stored after successful execution of the Twitter Server and show the Performance related Statistics.

TwitterSimulator > Stats.txt

```
1   Performance Statistics
2
3   Total number of requests = 100012
4   Total server uptime = 71.990950 seconds
5
6   Type of request - No. of requests - Average time per request(ms)
7   Register - 1000 - 1.059165
8   Follow - 7054 - 0.196871
9   Tweet - 35456 - 0.617602
10  Retweet - 1348 - 1.913055
11  QueryHashtag - 27558 - 0.794240
12  QueryMention - 27596 - 0.793097
```