COP5615 - Distributed Operating System Principles

Project 4 Part 1 – Twitter Clone and Client Tester/Simulator

Submitted By: Parth Gupta, UFID: 91997064

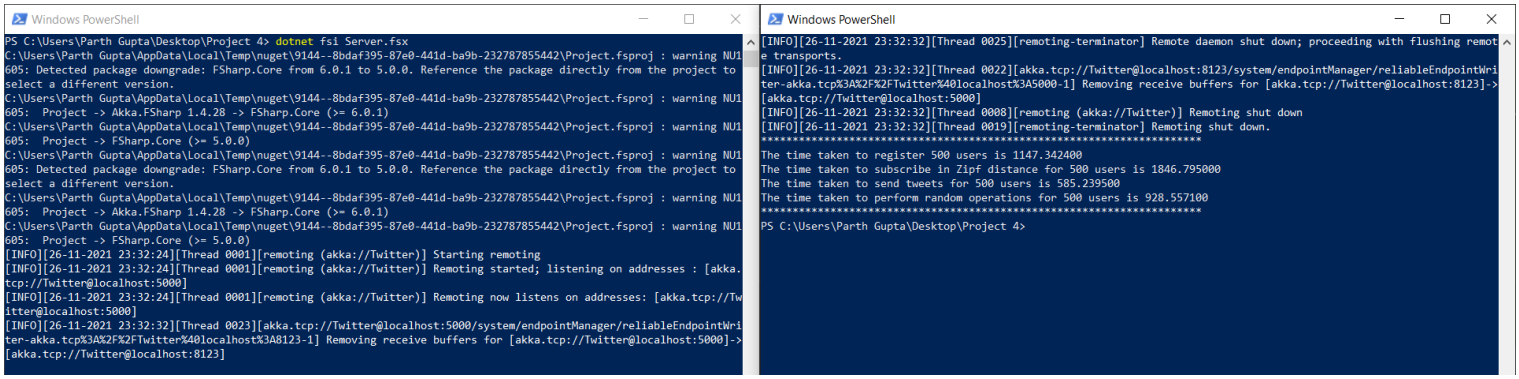
Instructions on How to Run the Code

1) Download the code from canvas submission.
2) After going to the folder where the file is downloaded. First run the command "dotnet fsi Server.fsx".
3) At the same time now run "dotnet fsi Client.fsx <numberOfClients>"

Input:

1) dotnet fsi Server.fsx
2) dotnet fsi Client.fsx 500

Output:



Problem Description

1) Implemented all the functionalities for a Twitter like engine that (in part II) will be paired up with WebSocket's to provide full functionality.
2) Some of the implemented functionalities of the engine are as follows:
   - Register User.
   - Send a Tweet. Tweets may have hashtags (like #COP5615isgreat) and mentions (@bestuser).
   - Subscribe to user's tweets.
   - Re-tweets (so that your 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).

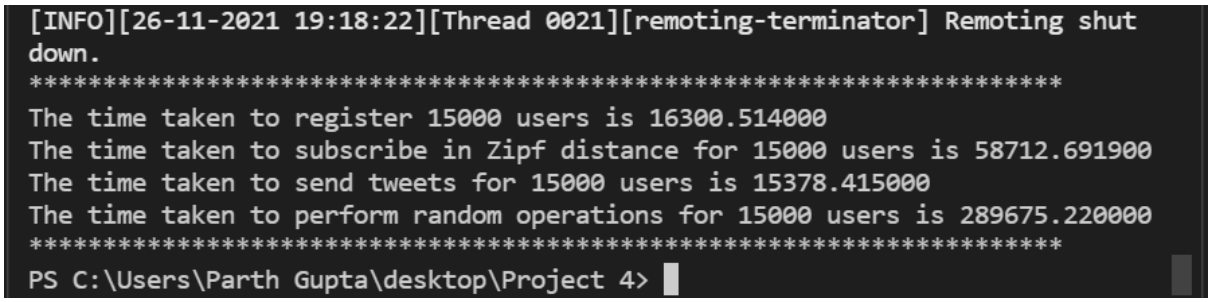
3) Implemented a Tester/Simulator to test all of the above functionalities.
    - Simulate as many users as you can.
    - Simulate periods of live connection and disconnection for users.
    - Simulate a Zipf distribution on the number of subscribers. For accounts with a lot of subscribers, increase the number of tweets. Make some of these messages re-tweets.

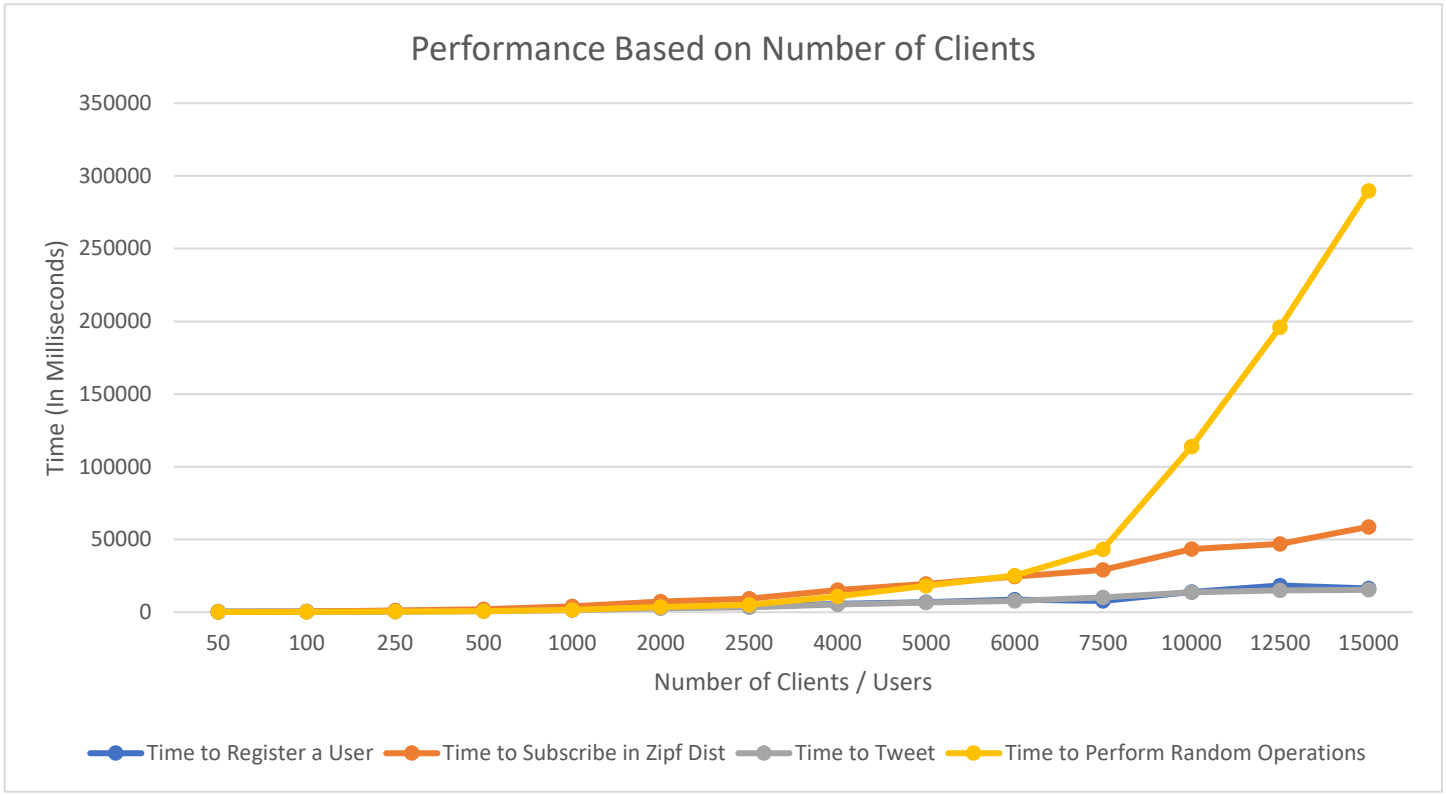
Performance Testing

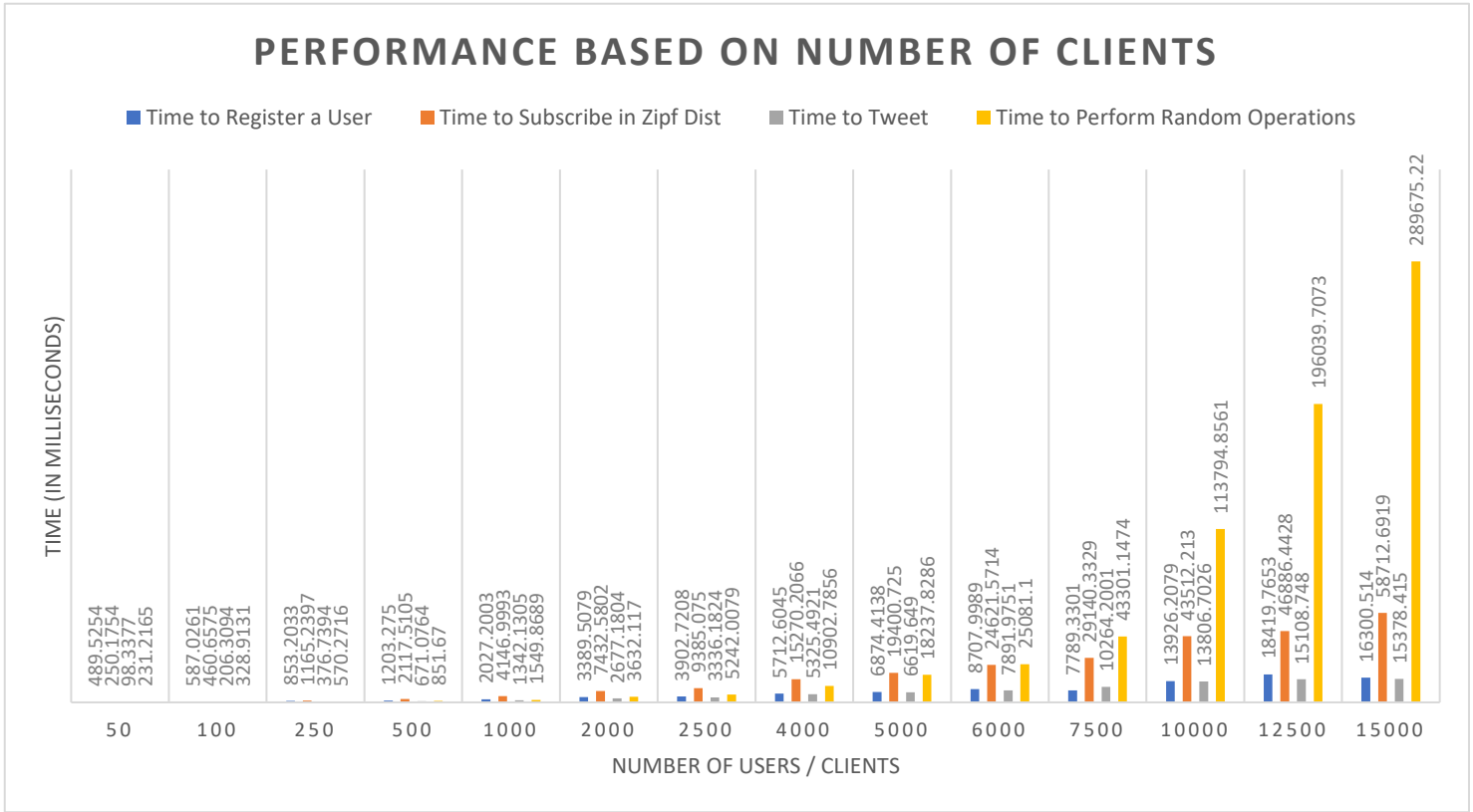
1) The clients are registered on the basis of input provided as a command line arguments.
2) After that I am subscribing follower to each client on the basis of Zipf distribution or Zipf Distance. Zipf = numberOfUsers / I^2. Where I belongs to [1, numberOfUsers].
3) I am assuming that each of the client is tweeting.
4) After that each client is performing N random operations which includes retweets, get mentions, get hashtags, query hashtags, etc.
5) After that I am printing time taken by each of the above activity and then measuring the performance by plotting the graph (time taken by these operations Vs Number of Clients).
6) The time taken to register a user and the time taken for an initial tweet took similar time in my implementation.
7) The time taken to perform random operations took the highest amount of time and after 6000 users, the time increased exponentially.
8) Simulating the subscribers using Zipf distance and the time taken to perform random operations took similar time up till 6000 users.
9) Time(Random Operation) > Time(Subscribing using Zipf Distance) > Time (Registering a User) = Time (Initial Tweet) [Approx.]
10) For this project, the largest network that I was able to manage was for 15000 users.

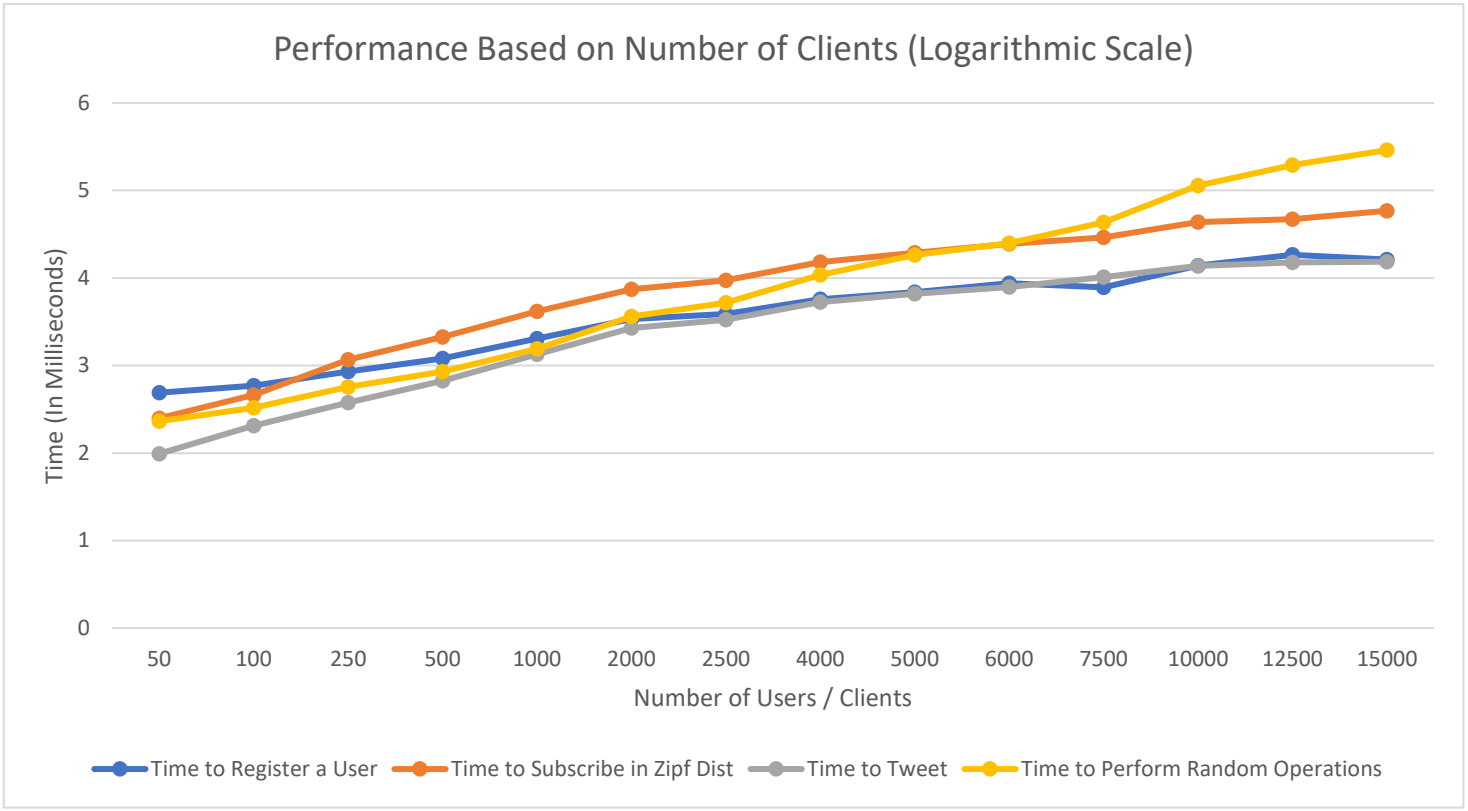
Time in Milliseconds

Number of Clients	Time taken to register a user	Time taken to subscribe in Zipf Dist.	Time taken to tweet initially	Time taken to perform random operations (tweet, retweet, etc)
50	489.5254	250.1754	98.3377	231.2165
100	587.0261	460.6575	206.3094	328.9131
250	853.2033	1165.2397	376.7394	570.2716
500	1203.275	2117.5105	671.0764	851.67
1000	2027.2003	4146.9993	1342.1305	1549.8689
2000	3389.5079	7432.5802	2677.1804	3632.117
2500	3902.7208	9385.075	3336.1824	5242.0079
4000	5712.6045	15270.2066	5325.4921	10902.7856
5000	6874.4138	19400.725	6619.649	18237.8286
6000	8707.9989	24621.5714	7891.9751	25081.1
7500	7789.3301	29140.3329	10264.2001	43301.1474
10000	13926.2079	43512.213	13806.7026	113794.8561
12500	18419.7653	46886.4428	15108.748	196039.7073
15000	16300.514	58712.6919	15378.415	289675.22




Performance Based on Number of Clients
Time (In Milliseconds)
350000
300000
250000
200000
150000
100000
50000
0
50
100
250
500
1000
2000
2500
4000
5000
6000
7500
10000
12500
15000
Number of Clients / Users
Time to Register a User
Time to Subscribe in Zipf Dist
Time to Tweet
Time to Perform Random Operations


PERFORMANCE BASED ON NUMBER OF CLIENTS
Time to Register a User
Time to Subscribe in Zipf Dist
Time to Tweet
Time to Perform Random Operations
TIME (IN MILLISECONDS)
489.5254
250.1754
98.3377
231.2165
587.0261
460.6575
206.3094
328.9131
853.2033
1165.2397
376.7394
570.2716
1203.275
2117.5105
671.0764
851.67
2027.2003
4146.9993
1342.1305
1549.8689
3389.5079
7432.5802
2677.1804
3632.117
3902.7208
9385.075
3336.1824
5242.0079
5712.6045
15270.2066
5325.4921
10902.7856
6874.4138
19400.725
6619.649
18237.8286
8707.9989
24621.5714
7891.9751
25081.1
7789.3301
29140.3329
10264.2001
43301.1474
13926.2079
43512.213
13806.7026
113794.8561
18419.7653
46886.4428
15108.748
196039.7073
16300.514
58712.6919
15378.415
289675.22
50
100
250
500
1000
2000
2500
4000
5000
6000
7500
10000
12500
15000
NUMBER OF USERS / CLIENTS


Performance Based on Number of Clients (Logarithmic Scale)
Time (In Milliseconds)
6
5
4
3
2
1
0
50
100
250
500
1000
2000
2500
4000
5000
6000
7500
10000
12500
15000
Number of Users / Clients
Time to Register a User
Time to Subscribe in Zipf Dist
Time to Tweet
Time to Perform Random Operations