Project 4 Part 1-TWITTER SIMULATOR AND ENGINE

NAME Jyotik Parikshya

UFID 0577-1395

NAME Daye Kim

UFID 8971-8937

Specifications:

Client

- Each user is represented by an actor in the program.
- The users follow other users, and each user has the id of the users the user follows in the list of the list.
- After all users and their following users are set, each user starts sending tweets including its id, hashtag, and mention.
- In Hashtag and mention, the actor's id is randomly stored.
- When the actor logins, it sends a request to the server to get the tweet messages whose actors are followed by it.
- After the actors get all the messages, they logout.

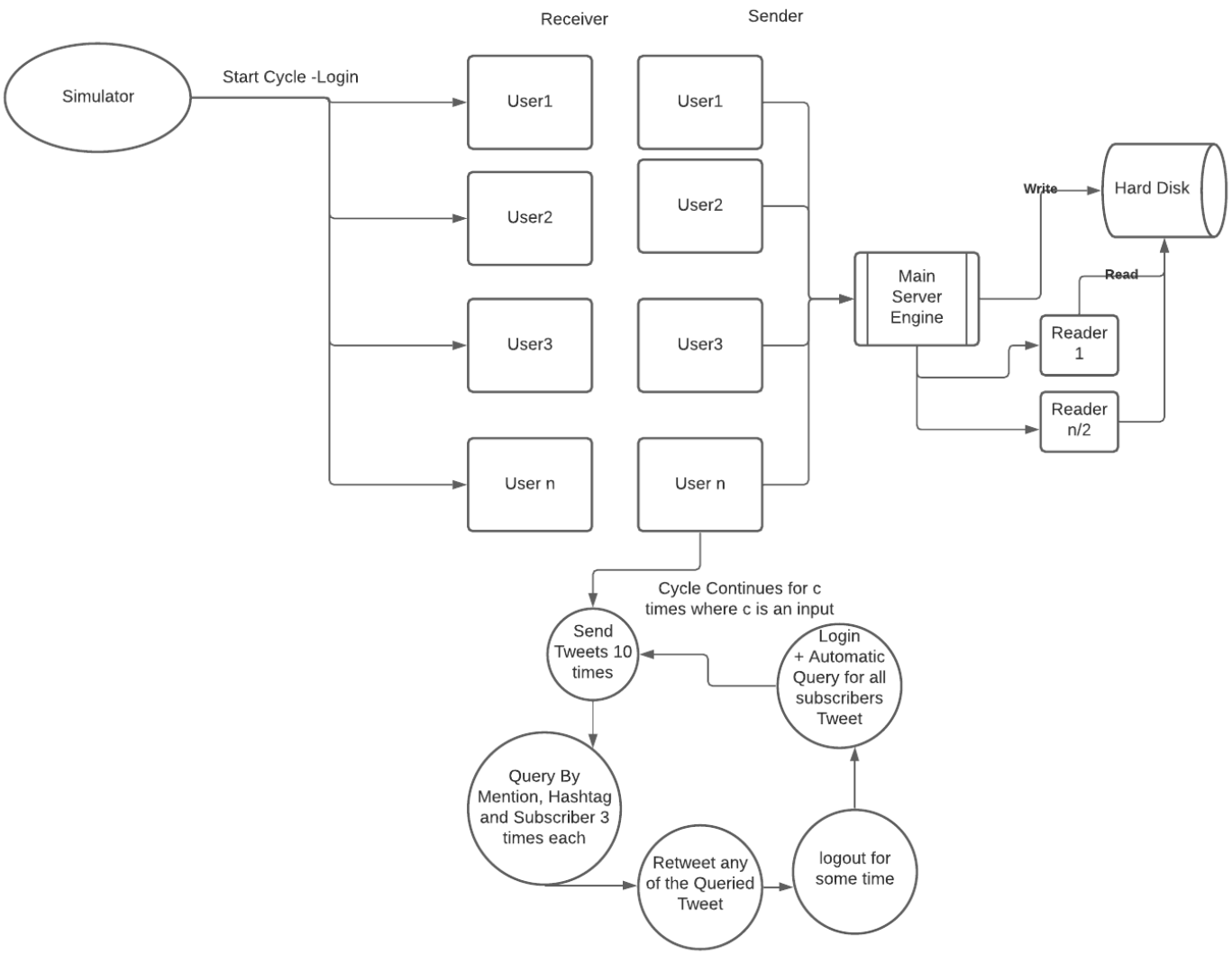
Server

- Server represents an engine
- It gets request messages from the user
- It sends back the requested tweets to each user
- It stores the data of users id, the following users of each user (Subscribers), and the tweets.
- Spawns Actors as Readers for concurrent Reading of data.

Data Base (In-Memory):

- Concurrent dictionaries were used to store Tweets (3 in total)
- List of List is used to store Subscriber data.
- Data Table (which is similar to Relation Data Base Tables) was initially used but it doesn't support concurrent read or write.

Structure of the Program:



Processes:

1) Server Side:

Number of actors Receiving message from Client: 1

Number of actors Reading from Data Base: Number of users/2

Number of actors Writing in Data Base: 1

2) Client Side:

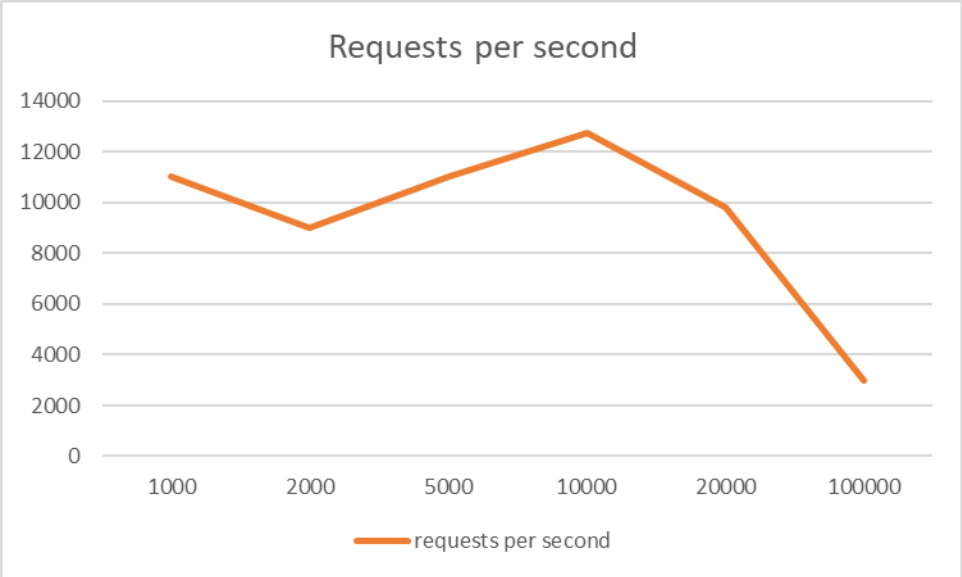
Number of Actors Receiving messages from simulator: Number of users

Number of Actors Sending Requests to Server: Number of users

Zipf Distribution:

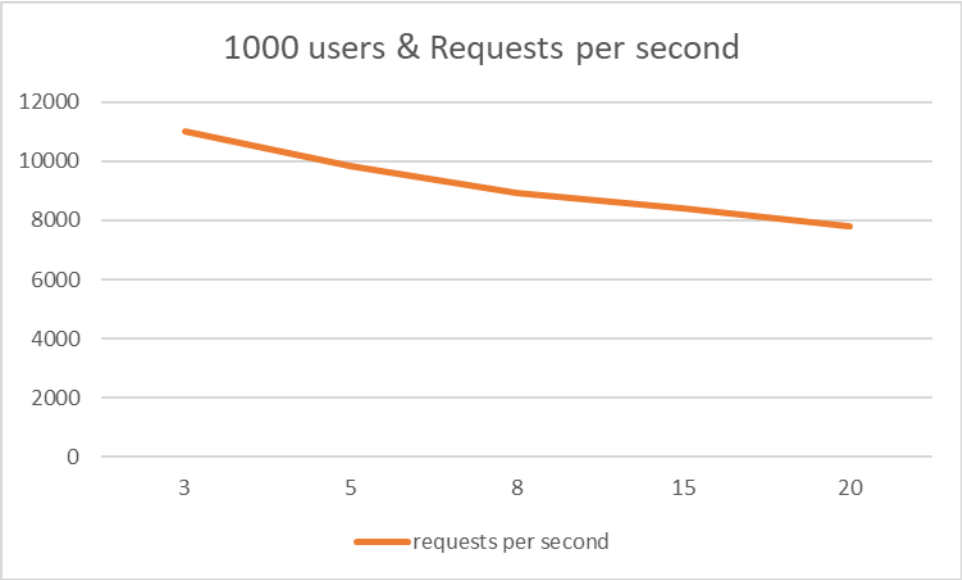
Zipf distribution is that the number of tasks is distributed by the given number divided by the rank. Then first ranked one will take the given number -1 of tasks and the second ranked one will take (the given number -1)/2 of tasks. The total number of tasks will be $\sum_{R=1}^{N} \frac{(T-1)}{R}$: T is the given number and R is the rank. In our project, each user is assigned ID in order of generation, and each user follows the different number of other users according to Zipf distribution.

Outputs and Observation:



Number of users	Cycles per user	Total Requests	Requests per second
1000	3	78,336	11005.12678
2000	3	133,700	9022.219248
5000	3	310,366	11035.61522
10,000	3	687,818	12724.62109
20,000	3	1,268,871	9823.55626
100,000	3	8,956,900	2965.176455

Each user sends three tweets and It terminates/ logouts when it gets the tweets from the other users who it follows. According to the result, within the same number of digits, the number of processed messages has less different, and the number of processed messages decreases when the number of users increases by one more digit.



Number of users	Number of cycles	requests per second
1000	3	11005.12678
1000	5	9835.959632
1000	8	8938.214508
1000	15	8415.002476
1000	20	7815.658222

The purpose of this experiment is to see that the number of cycles per user affects the number of processed messages per second. According to the output, the number of processed messages per second decreases when the number of tweets per user increases. The maximum number of messages is processed when the number of tweets is three. Basically, more the number of messages more load on the System and hence performance decreases.

Sample Screenshots:

The number of users :1000

```
G:\UF\Sem3\DOS\Twitter API\Part-1>dotnet fsi twitter_sim.fsx 1000 3
Server Built...Press Enter to start User simulatiuon..

Logging in and sending Tweets...
Real: 00:00:06.968, CPU: 00:00:12.921, GC gen0: 1067, gen1: 138, gen2: 1
Time Taken to converge: 7217 ms

Total Requests:78336
Requests per second:11005.126784
```

The number of requests messages per second: 11005.12678

The number of users :2000

```
G:\UF\Sem3\DOS\Twitter API\Part-1>dotnet fsi twitter_sim.fsx 2000 3
Server Built...Press Enter to start User simulatiuon..

Logging in and sending Tweets...
Real: 00:00:14.674, CPU: 00:00:30.156, GC gen0: 2243, gen1: 279, gen2: 1
Time Taken to converge: 14942 ms

Total Requests:133700
Requests per second:9022.219248
```

The number of requests messages per second: 9022.219248

The number of users :5000

```
G:\UF\Sem3\DOS\Twitter API\Part-1>dotnet fsi twitter_sim.fsx 5000 3
Server Built...Press Enter to start User simulatiuon..

Logging in and sending Tweets...
Real: 00:00:28.947, CPU: 00:00:50.921, GC gen0: 5913, gen1: 770, gen2: 1
Time Taken to converge: 29201 ms

Total Requests:310366
Requests per second:11035.615219
```

The number of requests messages per second: 11035.61522

The number of users :10,000

```
G:\UF\Sem3\DOS\Twitter API\Part-1>dotnet fsi twitter_sim.fsx 10000 3
Server Built...Press Enter to start User simulatiuon..

Logging in and sending Tweets...
Real: 00:00:54.011, CPU: 00:01:40.531, GC gen0: 11985, gen1: 1446, gen2: 4
Time Taken to converge: 54169 ms

Total Requests:687818
Requests per second:12724.621093
```

The number of requests messages per second: 12724.62109

The number of users :20,000

```
G:\UF\Sem3\DOS\Twitter API\Part-1>dotnet fsi twitter_sim.fsx 20000 3
Server Built...Press Enter to start User simulatiuon..

Logging in and sending Tweets...
Real: 00:02:08.982, CPU: 00:04:11.140, GC gen0: 23926, gen1: 3126, gen2: 3
Time Taken to converge: 129265 ms

Total Requests:1268871
Requests per second:9823.556260
```

The number of requests messages per second: 9823.55626

The number of users :100,00

```
G:\UF\Sem3\DOS\Twitter API\Part-1>dotnet fsi twitter_sim.fsx 100000 3
Server Built...Press Enter to start User simulatiuon..

Logging in and sending Tweets...
Real: 00:50:18.008, CPU: 00:32:13.843, GC gen0: 137755, gen1: 15949, gen2: 47
Time Taken to converge: 3022725 ms

Total Requests:8956900
Requests per second:2965.176455
```

The number of requests messages per second: 2965.176455

How to run:

dotnet fsi twitter_sim.fsx 1000 3

Imput1: Number of Users

Input2: Number of Cycles per user

After the Server is Running It will show a message and we have to press Enter to start the simulation process.