Facebook Simulator

Description:

- Akka Actor based client server model to support thousands of users
- A fully featured facebook simulator (client) and server (backend)
- An in memory server to process client requests
- REST API implemented in Spray Can

Features Implemented:

Client (Simulator) :

- Create thousands of users (Clients as Akka Actors)
- The users can perform the following actions :
    - Get Profile (friend/self)
    - Make a Friend
    - Get Friend List (anyone/self)
    - Post Message (to a anyone/self: with any privacy setting)
    - Get My Posts (own posts)
    - Get Timeline (anyone with privacy restrictions/self)
    - Post Picture (friend/self: with any privacy setting)
    - Get Pictures (friend/self: with any privacy setting)
    - Get Album (friend/self: with any privacy setting)

Server :

- Register Users
- Process incoming user requests and respond back to the client with requested data (JSON) over HTTP
- Save the following relevant user data in in-memory data structures :
    - Users' details
    - Posts
    - FriendLists
    - Pictures

Instructions to run the program:

Unzip the file:

- `$ cd FacebookProject.zip`
- `$ cd FacebookProject`
- `$ cd FB`

Server Configuration:

- `$ cd src/main/scala/Server`
- `$ vi Server.scala`
- Edit application configuration present in the Server file and replace the `http.host` field with IP address of the host (machine running the code):

```
http {
 host = "10.136.75.232"   <- replace
 host = ${?HOST}
 port = 5000
 port = ${?PORT}
}
```

To run the Server:

- Navigate back to the `FB` directory
- `$ sbt`
- `> run`
- Enter the option for `Server.Server` when prompted for `Enter number:`
- If successful the Server successfully binds to the IP and port specified in the application configuration and will wait for requests from any client

To run the Client:

- In a different terminal window:
- `$ cd FacebookProject`
- `$ cd FB`
- `$ sbt`
- `> run <host-server-IP> <host-server-port> <number-of-users>`
  Example: `> run 127.0.0.1 5000 10000`
- Enter the option for `Client.Client` when prompted for `Enter number:`
- If successful the Client starts sending requests to the server and starts displaying the response received from the server

Client Requests:

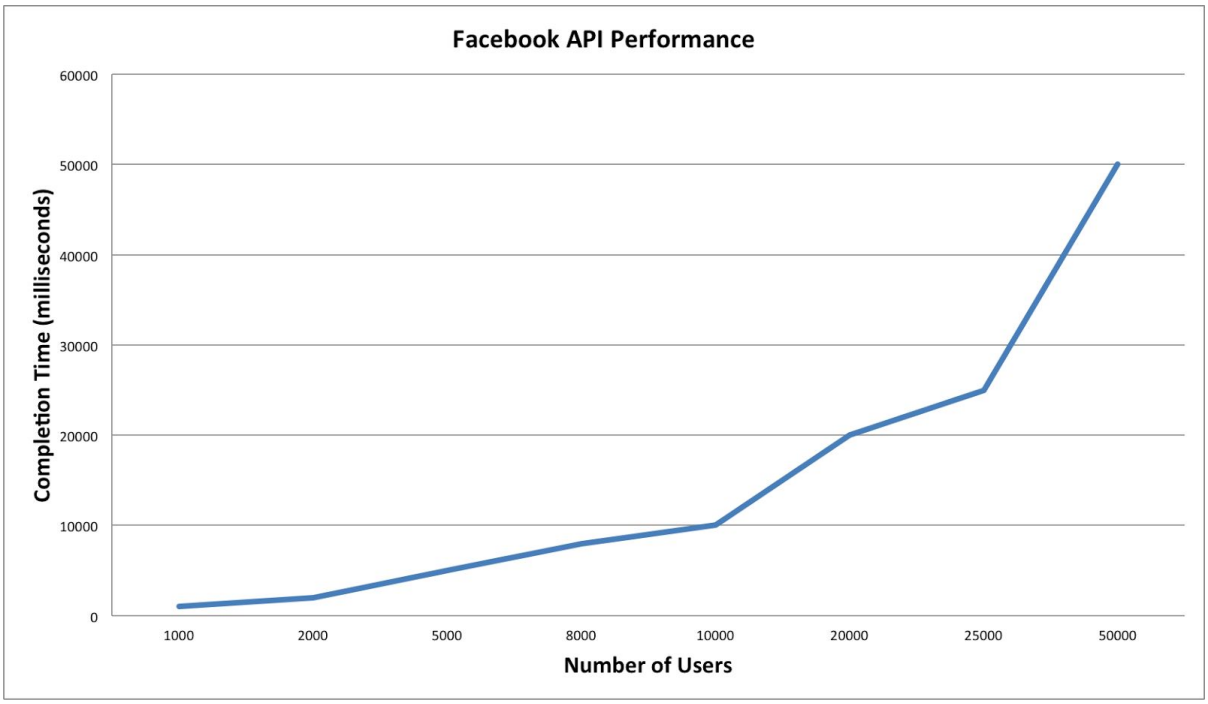
- For every user an Akka Actor instance is created
- As soon as the user is created, it is scheduled to execute the following messages for own/random users created so far:

```
self ! MakeFriend
self ! GetProfile
self ! GetFriendList
self ! PostMessage
self ! GetMyPosts
self ! GetTimeline
self ! PostPicture
self ! GetPictures
self ! GetAlbum
```

- These case classes construct the respective URI's and the corresponding GET/POST methods to hit the server with
- Once processed, the server responds back with the response status and/or the requested data if successful (POST requests receive just a response status; GET requests receive both response status and data)
- The execution finishes when all the actors have received all the responses back from the server

The below graph shows the performance of the code implemented in terms of the time taken v/s the number of users.

Maximum Number of Users Simulated : 125000



Number Of Users	Completion Time
1000	20335 milliseconds
2000	30748 milliseconds
5000	61504 milliseconds
8000	91594 milliseconds
10000	111963 milliseconds
20000	213040 milliseconds
25000	264078 milliseconds
50000	567751 milliseconds