COP5615: Fall 2015

PROJECT - 4 - I

Facebook Simulation & Rest API

Project Members

Sirivella Ananda Kishore   UFID: 9951 5080   asirivella@ufl.edu

READ ME

The directory hierarchy is as follows:

```
Project4-I
        |+ FbookClient
                |+ src
                        |+ main
                                |+ scala
                                        |+ fbookClient.scala
                                |+resources
                                        |+ localapplication.conf
                |+ build.sbt
        |+ FbServer
                |+ src
                        |+ main
                                |+ scala
                                        |+ fbServer.scala
                                |+resources
                                        |+ localapplication.conf
                |+ build.sbt
```

How to execute:

Input: The input provided (as command line to fbServer.scala & fbookClient.scala) will be of the form:

fbServer.scala : sbt “run”

fbClient.scala : sbt “run numUsers numPages”

numUsers: *Number of users to spawn.*
numPages: *Number of Pages to Create.*

The project deals with implementation of a facebook simulator and rest server.

fbServer.scala: Has a REST Server architecture to store data in-memory and respond with the operations data on call.
fbookClient.scala: The Client Server which simulates the behaviour of real-time facebook users and some of the facebook page functionality.

Part I : Facebook Simulator:

Similar to Bitcoin mining and Gossip algorithm akka implementation, we have utilize the akka actor model for the simulation. We will be spawning Users & pages as the actors. User Actor functionality includes:

- Creating a New User
- Updating & storing user profile details
- Adding Friends to the User and retrieving friend list details
- Creating, updating and deleting a post
- Creating, posting and following facebook pages

We have utilized self-made Statistics, by assuming yound age groups (18-24) being most active and old age group (40+) being least active. Being a concurrent system all the actor parallelly try to add friends, create posts and update profile detail. All the functionality of storing details and their map is taken care by REST API server.

Part II : REST API:

In the second part of the project, we have implemented a REST API server to store and support facebook functionality. We have utilized SPRAY.IO to spawn actors for each request. As per requirement, we had utilized in-memory for storing and retrieving of any data. The implementation of all the services need to actively handle a user profile, his/her post, friend list and facebook page activities.

Sample Service:

POST

URL: http://localhost:8000/createUser

Input: { “userName” : “Actor” }

Output: userId

** Most Services are accessible with ease using REST tools.*

Statistics Utilized:

AgeGroup	Percentage User of total population	Average Number of Friends
18 - 24	35%	1050
24 - 30	30%	900
30 - 40	20%	700
40 >	15%	550

Gender	Percentage Users
Male	45%
Female	55%

Observations:

- Was able to spawn 50,000 user at maximum with a post frequency of 500 milliseconds