COP 5615 - PROJECT 4 (PART -1)

PROJECT REPORT

FACEBOOK SIMULATOR

ARCHITECTURE:

The spray architecture has been implemented. The Facebook Simulator has been designed to send requests from client actors to the server actors via an HTTP listener actor bound to the local host IP 127.0.0.1 on port 8080.

FUNCTIONALITIES IMPLEMENTED:

The following functionalities have been implemented, and the information is saved on the server-side. The corresponding GET/POST requests have been mentioned:

Profile:

- create(userId,fname,lname,age,sex,location,occupation,interestedIn):
  This functionality is called by a POST request sent from the client with path 'create' with user id, first name, last name, age, sex, location, occupation and gender interested in as the parameters to the call.
- viewProfile(userId):
  This functionality is called by a GET request sent from the client with path 'viewProfile' and the user Id of the user whose profile is to be viewed as the parameter. The reply is in the form of a JSON object containing information like user id, first name, last name, age, sex, location, occupation, gender interested in, posts, friends, and pages liked.

Page:

- createPage(userId,pageId,desc):
  POST request with 'createPage' path and user id(the user creating the page,also the owner of the page), page id and a short description/title of the page as parameters.
- subscribeToPage(userId,pageId):
  POST request with 'subscribeToPage' path and user id (the user who wants to subscribe) and page id(of the page to be subscribed).
- listPages():
  GET request with 'listPages' path. The response is a JSON object with the page id, title of the page,owner's name and no. of followers of the existing pages.
- viewPage(pageId):
  GET request with viewPage as path and pageId to be fetched as parameter. JSON object in response has the pageID, title, posts , owner, subscriber details for the particular page.

Friend List:

- addfriend(friend1,friend2): POST request which adds the two friends whose ids are passed in the parameters with the path as 'addfriend'.
- getfriends(userId): POST request which fetches the list of friends and their details for user with user id passed in parameters and the path 'getfriends', in a JSON object.

Post:

- post(userId,message) : POST request which is sent when a user wants to update his/her status. Path for this request is 'post' with the user id and message in the parameters.
- postToPage(userId,pageId,message): POST request with path as 'postToPage' and parameters as user id, page id and message. This request is generated when a user wants to post to a page which he/she has subscribed to. Unless subscribed, the user cannot post to the page.
- postToUser(userId,friendId,message): POST request sent when a user wants to post a message to a friend's timeline. A notification is also sent to the friend, notifying him about the post on his/her timeline. Path 'postToUser' and parameters(userId, friendId, message).
- getposts(userId): GET request which fetches the posts on the wall of a particular user in JSON format. Path 'getposts' and parameters(userId).

Statistics:

- getStats(): Every time a user is added, a page is created or a message is posted, counters are incremented server-side and getStats GET request is scheduled to be sent periodically from the client to fetch these refreshed statistics on the client machine in a JSON format. Path: 'getStats'.

Miscellaneous:

- stopSystem(): POST request which triggers the system shutdown after 60 seconds of runtime. The server calls the GetStats function of the listener actor and the results are written onto an HTML file, which is later opened in the browser.
- getNotifications(userId): GET request which gets the notification for the user id passed user id in a JSON object.
- listUsers(): GET request which returns a JSON object listing the name, user id of all the existing users in the database.

System Shutdown:

The system runs for 1000 users for a duration of 60 seconds. After 60 seconds, the getStats GET request is sent to server and the client system is shutdown. On the server, the GetStats function is called on the listener actor and the server statistics in the Future returned is enveloped in a HTML format and written onto a file and this HTML page is then opened in the browser.

USER BEHAVIOR SIMULATION:

According to a study(referenced), ~35% of the users are in the age group of 25-34. This data has been used to construct the age database.

The users have been equally distributed among Very Active, Intermittently Active, and Dormant users.

According to a study, the order of type of posts is Status Update(most likely), post to friend's timeline(less-er likely than status updates), posts to subscribed pages(least likely).

- For Very Active users => status updates are scheduled every 300ms.

  posts to friends' walls are scheduled every 700ms.

  posts to subscribed pages' walls are scheduled every 1000ms.

- For Intermittently Active users => status updates are scheduled every 1500ms.

  posts to friends' walls are scheduled every 3500ms.

  posts to subscribed pages' walls are scheduled every 5000ms.

- For Dormant users => status updates are scheduled every 3000ms.

  posts to friends' walls are scheduled every 7000ms.

  posts to subscribed pages' walls are scheduled every 10000ms.

These post commands have been scheduled to run periodically in the client actor, while the actor is alive.

According to a study, the maximum number of users is typically ranging about 35% of the users considered for the study. Hence, in the program, the number of outgoing addfriend post commands is limited to max 35% of the total number of users.

RUNTIME PROGRAM FLOW:

For each client actor, the runtime flow is : create user => based on user id, let the user create a page => add random users as friends. After this initial setup, the schedulers for the posts takeover and users start posting messages.

RESULTS:

The data for name, occupation, location, ages was picked from text files in the resources of the project.

The system was run for 1 server actor and 1002 client actors for a duration of 60 seconds. The total number of pages created are 400. The **average number of requests processed by the server was 6000 per second**.

INSTRUCTIONS FOR RUNNING:

- Unzip the file to a specified directory and change the working directory.
- Open the terminal with 2 windows. Run the 'sbt run' command on both the windows
- Select the option to run server on one window and client on the other. Make sure you run the server first.

REFERENCES:

- Age distribution among the users: (age group distribution ~ 30% in the range 25-34):

https://zephoria.com/top-15-valuable-facebook-statistics/

- The avg. number of pages liked by a user is 40:

http://www.adweek.com/socialtimes/how-many-pages-does-the-average-facebook-user-like/418322