Distributed Operating Systems
Project 4 Part 1
Project Report

Sravya Pasupuleti

Niharika Vithala

Statistics Used For Building Facebook Simulator

Total number of users – 1.55 Billion

Number of daily active users – 1.35 Billion (87%)

Average number of friends per user – 190

Facebook friend distribution

Percentage of users	Number of friends
10%	<10
20%	<25
10%	25-100
40%	100-500
20%	>500

Facebook Friend distribution by age

Age	Number of Friends
ages 12,17	521
ages 18,24	649
ages 25,34	360
ages 35,44	277
ages 45,54	220
ages 55,64	129
ages 65+	102

- 35% of user's friends see a post on average.
- Average number of feed on a user's wall each day is 1500.
- Every 60 seconds on Facebook: 510 comments are posted, 293,000 statuses are updated, and 136,000 photos are uploaded.
- An average Facebook user has 217 photos.

We have scaled down the original values to manage the load on simulator.

Implementation Details

- We have simulated client and server which follow REST API.
- Server has been implemented using Spray.
- All the requests from client are sent to the SprayServer (Server.scala) which receives the http requests and delegates the work to 5 server actors (MainServer.scala).
- The Server actors process the http requests and return response to the client in JSON format.
- We use Vector to store the Page Entry and Profile Entry.
- As per our implementation : Page corresponds to News Feed in Facebook where in posts from the user and their friends appear
- As per our implementation : Profile contains some basic information like first name, last name and age and the posts made by the user and the posts wherein other users have mentioned a particular user. This is very similar to the timeline
- We have used random String to represent a picture. Data structure we have used for picture is Array of List of String
- Similarly for Albums we pair up existing pictures of a user to form albums. Thus Albums have been implemented using Array of List of List of Strings. This was fairly new for us and we learnt a lot from it
- Friend list has been implemented using Array of Integers
- In Client we have used spray-client where in http requests are sent using pipeline
- In Server we have tried to initialize friend list in such a way that it simulates actual Facebook statistics for how many friends users have
- On Client side, we have used System scheduler to generate requests based on heavy and light load as well as varying frequency of requests for Page, Profile, Picture, Album

Challenges in implementation

- Certain warnings and log errors are shown when we run the client. However, they do not impact the functioning of the simulator.
- Initially we used nested for loops for mapping friends to users. When the number of users was large (1000000) the process would go into an infinite loop. To speed up the process we have explicitly divided the process into smaller subroutines and implemented the friend distribution according to statistics.
- When the number of posts and photos uploaded is high, OutOfMemory error is thrown due to Java Heap space problem. Also GC overhead limit exceeded error is produced. We have reduced the number of posts and photos accordingly
- Server receives posts which are updated in the Page and Profile of the relevant users and Server receives requests like Get Friend list of a particular user, Get Photo of a particular user, Get Album of a particular user, View my news feed and View my profile

Results of implementation

We have done testing on a single 4 core machine. There have been logger warning and information messages which appear which we were unable to turn off.

Load Type	Heavy
Number of posts	12191
Number of requests	19459
Total	31650
Duration of run	600 seconds
Rate	52.75

Load Type	Light
Number of posts	20365
Number of requests	4490
Total	24855
Duration of run	600 seconds
Rate	41.425

Execution Details

Executing using sbt

To start the server:

1) Open a new terminal.
2) Navigate to the directory containing Server.
3) Execute the following command sbt “run <no_of_user>”
4) Once you see “Facebook server is up and running “ you can start the client.

To start the client:

1) Open a new terminal.
2) Navigate to the directory containing Client
3) Execute the following command sbt “run <no_of_users> <load_type>”.
4) Load type can be heavy/light.
5) Please ignore the logging messages that appear on the client terminal.

Specific details and project structure is given in the Read me file attached