Project 4
Design of Facebook Like REST API Server and Highly Parallel Request Client Simulator

Chunyang Shen 65499240

1. Facebook Like Server

1.1 Data Structure

In this system we store every object (user, public page, album, photo) as a node. The node contain fields and edges. Fields are static attributes like name, created time, description, etc. Most of fields are key-value pair. Edges are the relationships to other nodes. For example, for user node, it has a kind of edge named “Friends”, the value of this kind of edge is a HashSet, which contains the user id of his friends. It also has a kind of edge named albums, the value of this kind of edge is the id of his albums. So Edges are pairs of key-hashset. By visiting edge, we can access to other nodes related to the current one.

1.2 Store System.

All data are stored in Memcache. The Memcache is actually a actor contained a large hash table. The key of this hash table is the id of each node, and the value is the node object itself. It support methods like Add, Get, Add edge and Update. Since these methods are realized by message delivering and they are serial so no lock is need.

1.3 Highly Parallel Search Engine

The core search engine in this system is Finder. Finder is also a actor which receive search query and respond JSON like information. The query are in tree structure. After get the node according the top id of the query, the finder will get the first level information by the fields of a node, and exploring edges by generate new finders to continue search the second level information according the node id in edges. After this sub finder respond to the father finder, the father finder will gather all the information, transform it into JSON and respond the upper-level asker. After this process, the finder will kill itself. Future technique is used in Finder to ensure non-block and high parallel performance.

1.4 Highly Parallel REST Request

Spray Framework is used to support REST API. For each request, the Listener actor will generate an Operate actor to take responsibility to this request. In this way the listener can keep on listening new request. This mechanism can ensure highly parallel request.

2. Client Simulation

The Client will simulate 1000 users online simultaneously with different frequency of post and get. Request can be randomized REST API to post and get message, albums and photos.

3. System Support REST API

build profile (user):
```
  http://127.0.0.1:8080/profiles?name=[\w*]&birthday=[\d*]
```
build page (public page):
```
  http://127.0.0.1:8080/pages?name=[\w*]&createdate=[\d*]
```

build album:
  http://127.0.0.1:8080/<user-id>/albums?name=[\w*]&discription=[\d*]
post photo:
  http://127.0.0.1:8080/<user-id>/albums/<album-id>/photos?name=[\w*]&discription=[\d*]
post text
  http://127.0.0.1:8080/<user-id>/posts?title=[\w*]&content=[\w*]
get profile
  http://127.0.0.1:8080/<profile-id>
get page
  http://127.0.0.1:8080/<page-id>
get post list
  http://127.0.0.1:8080/<user-id>/posts
  http://127.0.0.1:8080/<user-id>/posts?title&content
get friend list
  http://127.0.0.1:8080/<user-id>/friends
  http://127.0.0.1:8080/<user-id>/friends?name&birthday
get album list
  http://127.0.0.1:8080/<user-id>/albums
  http://127.0.0.1:8080/<user-id>/albums/name&discription
get photo list in a album
  http://127.0.0.1:8080/<user-id>/albums/<album-id>/photos
  http://127.0.0.1:8080/<user-id>/albums/<album-id>/photos?name&discription
get photos in album lists
  http://127.0.0.1:8080/<user-id>/albums//photos
  http://127.0.0.1:8080/<user-id>/albums//photos?name&discription
get specific album
  http://127.0.0.1:8080/<user-id>/albums/<album-id>/
  http://127.0.0.1:8080/<user-id>/albums/<album-id>?id&name&discription
get specific post
  http://127.0.0.1:8080/<user-id>/posts/<post-id>/
  http://127.0.0.1:8080/<user-id>/posts/<post-id>?id&title&content
get specific photo
  http://127.0.0.1:8080/<user-id>/albums/<album-id>/photos/<photo-id>/
  http://127.0.0.1:8080/<user-id>/albums/<album-id>/photos/<photo-id>/?id&name&discription
get specific profile of a friend
  http://127.0.0.1:8080/<user-id>/friends/<friend-id>
  http://127.0.0.1:8080/<user-id>/friends/<friend-id>?id&name&birthday

4. How to Run

Goto Server and Client folder using “sbt run”