Entry Task - Shared Service

Usage of the official `net/rpc` package or `gRPC` is not allowed.

1. Requirements

Features

Implement a user manager system for user to login and edit their profiles. Users can login on web page.

After a user inputs his/her username and password, the backend system will authenticate his/her identity. If the login is successful, the relevant user information is displayed, otherwise an error message will be shown.

After a successful login, a user can edit the following information:

1. Upload a picture as his/her profile picture
2. Change his/her nickname (support unicode characters with utf--8 encoding)

User information includes: username (cannot be changed), nickname, profile picture. For test, the initial user data can be directly insert into database. Make sure there are at least `10 million` user accounts in the test database.

Time Limit

You need to finish this task in 5 working days.

Performance

- Supports **1000** concurrent http requests
- With cache layer,
    - Basic requirement:
        - supports **6000** login requests per second
        - from at least **200** unique users
        - no socket/timeout error
    - supports **20k** login requests per second (Optional, you can try reach this target if have time)
- Without cache layer,
    - Basic requirement:
        - supports **3000** login requests per second
        - from at least **200** unique users
        - no socket/timeout error
    - supports **10k** login requests per second (Optional)

Stress Test Environment

- Server: VM environment, 8 CPU cores 8G memory. You can apply via SPACE
- OS: Ubuntu 16.04
- DB: MySQL 5.7
- Client: Chrome and Firefox

1. If you can't open SPACE, try connect to VPN.
2. After you raise GTS/GAC ticket to apply permission, reach your mentor for faster approval.

Design Constraint

- Separate HTTP server and TCP server and put the main logic on TCP server
- Backend authentication logic should be done in the TCP server
- User information must be stored in a MySQL database. Connect by MySQL Go client.
- Use standard library whenever possible. Build the connection pool library yourself.
- Web interface will not directly connect to MySQL. For each HTTP request, web interface will send a TCP request to the TCP server, which will handle business logic and query the database.

Design Considerations

- Robustness

- Security
- Performance

2. Deliverables

- Project source
- Design document
- Installation and maintenance documentation
- Performance tests report
- Concluding report

3. References

- Go: http://golang.org
- Coding style: https://github.com/golang/go/wiki/CodeReviewComments
- Testing: https://golang.org/pkg/testing/
- Profiling: http://blog.golang.org/profiling-go-programs
- Go Web application example: https://golang.org/doc/articles/wiki/
- Go editor/IDE
    - https://www.jetbrains.com/go/
    - https://code.visualstudio.com/
    - https://github.com/fatih/vim-go
    - https://github.com/dominikh/go-mode.el
    - https://github.com/DisposaBoy/GoSublime
    - https://github.com/visualfc/liteide
- MySQL Golang library: https://github.com/go-sql-driver/mysql
- MySQL Client: http://sequelpro.com/
- Redis: http://redis.io
- Redis Client: https://github.com/go-redis/redis
- Protobuf: https://developers.google.com/protocol-buffers
- Database Design Guide

4. Performance Leaderboard

#	QPS(without cache)	At
1		
2		
3		