Project 4: Reddit Clone in GO - Part 1 & 2

Course: COP5615 – Distributed Operating System Principles

Group Members:

Chenna Kesava Vara Prasad Korlapati (UFID: 4836-8778)

Phalguna Peravali (UFID: 3753-9361)

Overview:

This report presents a comprehensive overview of Reddit Clone in GO, developed for the COP5615 course. The project implements a Reddit-like engine with core functionalities, a client simulator, and a REST API interface, showcasing the application of concurrent programming principles using the Go programming language. The system demonstrates key features of a social media platform, including user registration, content creation, voting, and messaging, all implemented using the Actor model for concurrency and a REST API for web client integration.

Project Objectives:

The primary objectives of this project were to:

1. Implement a Reddit-like engine with core functionalities.
2. Develop a client simulator to test the engine's performance.
3. Demonstrate the application of concurrent programming techniques.
4. Analyze system performance under various user loads.
5. Implement a REST API interface for the engine.
6. Create a simple client that uses the REST API
7. Demonstrate functionality with multiple clients.

System Architecture:

The project is structured into 6 main components:

1. **Engine (Engine/engine.go)**: The core component implementing Reddit-like functionalities using the Actor model.
2. **Simulation (simulation/simulation.go)**: A module that simulates multiple concurrent users interacting with the engine.
3. **Main (main.go)**: The entry point that initializes the ActorSystem and orchestrates the simulation.
4. **API (api/api.go):** Implements a REST API interface for the engine using the Gin web framework.
5. **Client (client.go):** A simple client implementation to interact with the API.
6. **Multi-Client (multi_client.go):** An implementation with two clients to interact with the API concurrently.

Key Features:

- User registration and management
- Subreddit creation, joining, and leaving
- Posting and commenting in subreddits
- Upvoting and downvoting posts
- Karma computation
- Direct messaging between users
- REST API for all core functionalities

Implementation Details:

4.1

Engine Module:

The engine is the main module of the system, handling all core functionalities:

Key methods in the Engine module:

- RegisterUser()
- SubredditSpecificOp()
- CreatePost()
- ReplyToComment()
- SendDMtoUser()
- ReplyToAllDMs()
- UpvoteRandomPost()
- DownvoteRandomPost ()

These methods implement the various features of the Reddit clone, managing user interactions, content creation, and voting mechanisms.

Simulation Module:

The simulation module creates multiple concurrent users who perform random actions on the Reddit clone:

Main simulation function:

SimulateUser(): This function orchestrates a series of random actions for each simulated user, including registering accounts, joining subreddits, creating posts, and sending messages.

Performance Analysis:

The system's performance was evaluated by simulating various numbers of concurrent users. The following table summarizes the key statistics:

Total Users	Total Posts	Total Subreddits	Total Messages	Total Simulation Time
10	131	106	113	40.5556424s
50	603	541	596	42.121259s
100	1186	1111	1141	43.4298415s
500	5407	5448	5323	44.0380639s
1000	10423	10779	10683	44.5763764s
5000	56089	56706	54101	45.0230307s

The results demonstrate the system's ability to handle increasing user loads with minimal impact on overall simulation time, showcasing the effectiveness of the concurrent design.

4.2

API Module:

The API module implements a REST interface using the Gin web framework:

Key endpoints:

- *POST /api/register* (Register a new user)
- *GET /api/user/:username* (Get user information)
- *POST /api/user/:username/join* (Join a subreddit)
- *POST /api/user/:username/leave* (Leave a subreddit)
- *POST /api/subreddit* (Create a new subreddit)
- *GET /api/subreddit/:name* (Get subreddit information)
- *POST /api/submi*t (Create a new post)
- *POST /api/posts/:id/upvote* (Upvote a post)
- *POST /api/posts/:id/downvote* (Downvote a post)
- *POST /api/comment* (Create a comment)
- *POST /api/message/compose* (Send a direct message)
- *GET /api/message/inbox* (Get user's direct messages)
- *GET /api/feed* (Get user's feed)

Client Module:

The client module provides a simple interface to interact with the API:

Key methods:

- RegisterUser()

- CreateSubreddit()
- JoinSubreddit()
- LeaveSubreddit()
- CreatePost()
- UpvotePost()
- DownvotePost()
- SendDirectMessage()
- GetUserInfo()
- GetFeed()

How to Run:

Start the Reddit-like engine server: *go run main.go*

Run the single client to check functionality of API: *go run client.go*

Run the multi-client simulation: *go run multi_client.go*

Results:

The multi-client simulation demonstrates the system's ability to handle concurrent requests from multiple clients using the API. It showcases various interactions such as user registration, subreddit creation, posting, voting, and messaging.

Demo Video:

Here's the link to the demo video for part 2: link

Conclusion:

The project successfully implements a Reddit-like platform using Go's concurrency features and provides a REST API for web client integration. The system demonstrates good scalability, handling thousands of simulated users and multiple concurrent clients with minimal increase in execution time. The addition of the REST API enhances the system's usability and allows for easy integration with web-based front end.

Future Improvements:

1. Implement user authentication and authorization.
2. Add support for images and markdown in posts.
3. Develop a more sophisticated karma system.
4. Implement a web-based front-end to interact with the API.
5. Add caching mechanisms to improve performance for frequently accessed data.

These enhancements would further improve the system's realism, security, and utility, making it more suitable for real-world deployment.

References:

1. Go Programming Language. https://golang.org/
2. Actor Model. https://en.wikipedia.org/wiki/Actor_model
3. Reddit. https://www.reddit.com/
4. Gin Web Framework. https://github.com/gin-gonic/gin
5. RESTful API Design. https://restfulapi.net/