Task 2: P1-T2 – Message boards

Due: 10/6/2017, 11:59pm CDT

Early bird deadline: 10/2/2017, 11:59pm CDT

WARNING: Read Project 1 MongoDB specification before starting this task

1. Provided information

The AggieFit team wants to add message boards to encourage the employees to interact and exchange information with each other. They intend to categorize the message boards by topics. The employees can view and add messages to boards that they are interested in.

The team took recommendation from experts and were told that they could use two databases based on their requirements: MongoDB and Redis. The experts were not told about the application. The AggieFit team wants to know where the databases fit in their message board application. The basic functionality they desire is as follows:

(i) Select message board: Choose a message board to read and write from. All operations are performed only after the message board is selected.
(ii) Read: Should be able to read all messages on the selected message board.
(iii) Write: Write the message into the selected message board.
(iv) Listen to updates: Waits to get real time updates on the selected message board. If any other user writes on this message board, you should be able to read it immediately. Unlike read, the user does not have to explicitly pull the data. If the user is listening on a board, they should receive updates immediately after the board is updated.

2. The tasks

P1-T2-a: The team asks you to propose a system architecture (similar to DropCam and Slack as you saw in the workloads video, but focusing on the databases). They also want you to implement a prototype for the message board application using your proposed architecture.

First, you will need to go through the features of both databases and identify which one is suitable for what part and propose where the databases fit in the architecture. The architecture should clarify what happens when a user selects, reads from, writes to, and listens to the message boards.

Next, you will need to implement a simple prototype that demonstrates basic functionality of the system. For now, a simple command line output is sufficient. Test scenarios will involve multiple users accessing the boards in a particular order. The following operations should be supported:

(i) Select message board: select <board_name>
(ii) Read: read. If no board is selected, output an error that no board was selected.
(iii) Write: write <message>. Output and error if no board is selected.
(iv) Listen to updates: listen. Output and error if no board is selected.
(v) Stop listening to updates: stop or <Ctrl-C>. Output and error if no board is selected and not listening.

Two sample scenarios are attached on eCampus->Content->Project 1.

For the basic case, you can assume that the different users of the system are time synchronized, and therefore see the same ordering of operations. We suggest you code the prototype without any error checking, i.e., assuming that all commands adhere to the specification. You can add error checking incrementally.

MongoDB download and installation instructions are already provided to you in task 1. For Redis, download and install using the instructions in https://redis.io/download. To interface with the programming language of your choice, use https://redis.io/clients. The desired library for python is redis-py. A sample script for a different application is provided along with the lectures.

P1-T2-b: OPTIONAL FOR BONUS POINTS: The team wants your input on the consistency level required for their application if they make the database distributed. Specifically, they would like to understand the tradeoff between consistency and availability for their application.

3. What your client wants you to do

You should do the following to complete this task.

You need to **prepare a report** that contains the following:

1. The time you took to complete this task
2. P1-T2-a:
    i. A section on your architecture. You may have decided to use the databases for different purposes. You should explain your choice in the report. Also, you will need to explain what happens when a user performs an operation on the system.
    ii. A section on your prototype and instructions on how to run your code.
3. P2-T1-b: OPTIONAL FOR BONUS POINTS
    i. A section with your comments on the consistency model for the application

4. Project submission and grading rubric

You submit a zip file containing your report and code. The submission "button" is located on eCampus -> Content -> Project 1. This task is worth 40 points. You are required to use github for your code. The principles used for grading will be as follows:

- Failure to use github: -40 points
- Missing system architecture: -20 points
- Missing instructions on how to run your code: -20 points
- Not explaining your design choices: -20 points
- Prototype fails: -10 points
- BONUS of 15 points for adding error checking on the input in your prototype
- BONUS of 10 points for P1-T2-b.

If you submit by the early bird deadline, you get 5 points extra.

Partial credit will be applied if the answers are not thorough.