Assignment 3 (125 Points)

Consider the Ivy Architecture discussed in the class. In the Ivy architecture, the central manager (CM) is in charge of scheduling the read and write requests to the appropriate owner of a page. However, the central manager (CM) captures a single point of failure. In this assignment, we will implement a fault tolerant version of the Ivy architecture using the GO language.

To consider fault tolerance in the design, assume that we implement a backup central manager. You will design a protocol to maintain the consistency of (meta)data between the primary and the backup CM. When the primary CM fails, the backup CM takes over. When the primary CM restarts, then the control is handed over to the primary CM again taking care of the (meta)data consistency. Note: the assignment does not demand/require the implementation of Paxos state machine.

Using the GO language, implement the following variant of Ivy architecture:

1. **(10 marks)** First implement the Ivy architecture discussed in the class.
2. **(15+15 marks)** Based on the aforementioned ideas, design and implement the fault tolerant version of the Ivy architecture (the primary and back replica for the CM with consistency + related changes in the basic Ivy protocol).
3. **(10 marks)** Discuss whether your fault tolerant version of Ivy still preserves sequential consistency or not.

Experiments (Measure the end-to-end time performance with multiple read/write req.):

(15 marks) Without any faults, compare the performance of the basic version of Ivy protocol and the new fault tolerant version using requests from at least 10 clients. In this question, the requests could be read or write, randomized.

(15 marks) Without any faults, compare the performance of the basic version of Ivy protocol and the new fault tolerant version using requests from at least 10 clients. In this question, evaluate the performance for read-intensive workload (e.g., 90% reads and 10% writes) and then for write-intensive workload (e.g., 90% writes and 10% reads).

(15 marks) Evaluate the new design in the presence of a single fault – one CM fails only once. Specifically, you can simulate two scenarios a) when the primary CM fails at a random time, and b) when the primary CM restarts after the failure. Compare the performance of these two cases with the equivalent scenarios without any CM faults.

(15 marks) Evaluate the new design in the presence of multiple faults for primary CM – primary CM fails and restarts multiple times. Compare the performance with respect to the equivalent scenarios without any CM faults.

(15 marks) Evaluate the new design in the presence of multiple faults for primary CM and backup CM – both primary CM and backup CM fail and restart multiple times. Compare the performance with respect to the equivalent scenarios without any CM faults.

Assignment Submission Instruction:

Please follow the submission instructions carefully. Note that we need to run your code and need to interpret the outputs generated by your code. Thus, it is important to strictly follow the submission instruction as follows:

1. Collect all submission files and compress them in a zip file. Name the submission of your homework as **PSET3_<your_student_id>.zip**
2. Include a README file in your submission that clearly explains how to compile and run your GO programs. As the homework demands different configurations in simulation, explain in your README clearly to distinguish the cases (for example, if you have any command line features or arguments, explain them clearly).
3. If you have used any external package for GO (should not be required for this homework), kindly explain them in your README. Note that you are still required to code the protocols yourself.
4. Kindly include some information in the README on how to interpret the output of the program. Note that we will also check the source code. Thus, even though it is not required to excessively comment, a comment per GO routine is appreciated. Basically, each GO routine can carry a comment on what it is supposed to do.
5. Your README should specify how to activate each scenario in line with the homework questions. For example, the question on the coordinator election has five different scenarios. It is OK, if you must handle them using different functions. However, clearly mention in the README about the flags that we need to pass (via the command line) to activate each scenario for evaluation.
6. Any other information that you think helpful for running your code (e.g., open issues, assumptions) will be appreciated too.