Northeastern University – Silicon Valley

CS 6650 Scalable Dist Systems

Project #3 Assigned: 2/29/20 Due: 3/15/20 [200 points]

Multi-threaded Key-Value Store using RPC

Guidelines

Project #3 should be electronically submitted to Blackboard by midnight on the due date. A submission link is provided on the course BB page.

Assignment Overview

For this project, you will extend Project #2 in two distinct ways.

1) Replicate your Key-Value Store Server across 5 distinct servers. In project #2, you used a single instance of a Key-Value Store server. Now, to increase Server bandwidth and ensure availability, you need to replicate your key-value store at each of 5 different instances of your servers. Note that your client code should not have to change radically, only in that your clients should be able to contact any of the five KV replica servers instead of a single server and get consistent data back from any of the replicas (in the case of GETs). You client should also be able to issue PUT operations and DELETE operations to any of the five replicas.
2) On PUT or DELETE operations you need to ensure each of the replicated KV stores at each replica is consistent. To do this, you need to implement a two-phase protocol for updates. We will assume no servers will fail such that 2 Phase Commit will not stall, although you may want to defensively code your 2PC protocol with timeouts to be sure. Consequently, whenever a client issues a PUT or a DELETE to *any* server replica, that receiving replica will ensure the updates have been received (via ACKs) and commited (via Go messages with accompanying ACKs).

As in project #1, you should use your client to pre-populate the Key-Value store with data and a set of keys. The composition of the data is up to you in terms of what you want to store there. Once the key-value store is populated, your client must do at least five of each operation: 5 PUTs, 5 GETs, 5 DELETEs.

Your implementation may be written in Java. Your source code should be well-factored and well-commented. That means you should comment your code and appropriately split the project into multiple functions and/or classes; for example, you might have a helper function/class to encode/decode UDP packets for your protocol, or you might share some logic between this part of the assignment and the TCP client/server in the next part.

Evaluation

Your newly replicated (with consistency via 2PC) multi-threaded Key-Value Store servers will be evaluated on how well they interoperate with each other using RPC while doing concurrent operations on the UWT-provided "cluster" as well as their conformance to the requirements above.

Executive Summary

Part of your completed assignment submission should be an executive summary containing an "Assignment overview" (1 paragraph, up to about 250 words) explaining what you understand to be the purpose and scope of the assignment and a "technical impression" (1–2 paragraphs, about 200–500 words) describing your experiences while carrying out the assignment. The assignment overview shows

how well you understand the assignment; the technical impression section helps to determine what parts of the assignment need clarification, improvement, etc., for the future.

Evaluation

The grade for your executive summary is based on the effort you put into the assignment overview and technical impression. In general, if you put some effort into your writing, you will receive full credit for your executive summary (provided that it is properly formatted and submitted as a plain text file).

Project Deliverables

The following items should be archived together, e.g., placed in a `.zip` file or tarball file (*.tgz or *.tar.gz), and electronically submitted via the link is provided on the course Moodle page.

1. All novel Java source code files implementing the two client and two server programs, i.e., plus any additional support code
2. Your executive summary.