CSCI555 Project Proposal

Team members: Chieh Nien, Junkai Liang

Project Overview

The evolution of distributed systems has been marked by the increasing complexity of modern applications and the need for scalable, reliable, and fault-tolerant infrastructure to support them. Distributed systems have transitioned from simple client-server architectures to highly distributed and decentralized environments, driven by the demands of web-scale applications and cloud computing. In this context, robust coordination and synchronization mechanisms are needed as as a pivotal component in the landscape of distributed systems. However, locks or mutexes that are commonly used on single-machines face severe challenges when applied to distributed systems.
We are curious about this problem. After our research, we found Google developed a locking service which ensures that multiple processes running on different machines coordinate with each other effectively, called Chubby.
So for our CSCI555 project, we decide to implement the Chubby system from the paper titled: 'The Chubby lock service for loosely-coupled distributed systems.' Chubby is a locking service and a small file storage system that provides coarse-grained locking and naming services and aims to solve the consensus problem in distributed systems.
When Chubby paper was published, Paxos was the pervasive consensus algorithm, so Google indeed utilized the Paxos to implement its consistency protocol in the paper. While, another consensus algorithm, Raft was designed in 2014 and was intended to be more understandable than Paxos while providing similar levels of fault tolerance. And we find that Chubby can be migrated to Raft, as well.
So we would like to implement a simple version of Chubby system based on the two consensus protocols - Paxos and Raft and compare the performance difference (described in Evaluation Section) between them. However, in consideration of time and feasibility, we have decided to use existing libraries instead of implementing these protocols from scratch. So that we can mitigate the impact of our implementation on performance, and meanwhile we can spend more time on figuring out the reasons that make the differences.

Evaluation

Our evaluation will be performed on both Paxos-based version and Raft-based version, and our goal is not only to evaluate the performance of Chubby system, but also to tell how these two consensus protocols make differences and the reasons. The metrics we plan to evaluate is listed as below.

- Performance of Chubby itself
    - Throughput without race conditions
    - Latency without race conditions

- Availability
  - Master server failure recovery
  - Slave server failure recovery
  - Client failure recovery
- Scalibity
  - Performance trend with increasing servers
  - Performance trend with increasing clients

First, we'd like to test the performance of Chubby itself, including throughput and latency. We will make locking and unlocking operations when there are no race conditions. So that we can get the overheads of the locking service itself.
Second, failure tolerance is the core part of distributed systems, so we want to make sure our system has high availability. We are going to emulate failures of master server, slave server, and clients, and observe the system states while recovery, respectively.
Third, we'd like to evaluate the scalibity of our implementation. We will test from two aspects, one is to increase the number of servers; the other one is to increase the number of clients connecting to the same amount of servers. We will analyze the performance trends of these two scenarios.

MileStones

Week	Time period	Jobs
1	Feb.12 ~ Feb.18	Fully understand Chubby / Raft / Paxos papers
2~3	Feb.19 ~ Mar.03	(1)Select foundational Raft and Paxos libraries for implementation (2)Get familiar with Golang
4~5	Mar.04 ~ Mar.17	Implement Chubby with Raft
6	Mar.18 ~ Mar.24	Write project intermediate report
7~8	Mar.25 ~ Apr.07	Implement Chubby with Paxos
9~10	Apr.08 ~ Apr.21	Evaluate on both versions
11	Apr.22 ~ Apr.28	Write project final report and prepare presentation

Colaboration Strategy

Our colaboration strategy is roughly half-half. Both of us will read the papers and discuss to fully grasp the concepts. Each of us will implement one version of the system and perform evaluation on it. We will together analyze the results and write the reports.