CS1380: Publish Path Caching and Log Compaction

Michael Li

May 8, 2023

For my work in CS1380: Distributed Systems, I implemented two additional features:

The first, publish path caching, was implemented for Tapestry, a decentralized distributed object location and retrieval system that uses prefix-based routing. Under a normal implementation, when publishing a key-value pair, each node forwards the pair to a root node; when looking up a key, each node calls findRoot to get closer to the key's root, which is the only node that stores the key-value pair. With publish path caching, when publishing a pair, each node instead publishes the pair to its store before forwarding. When looking for the key, then, each node can check if it has the key's information and return the pair immediately if it does, saving time and increasing availability.

The second, log compaction, was implemented for Raft, a consensus algorithm. Without this extra feature, each node's log can grow without bound, causing memory and time issues that can lead to availability problems. With log compaction, the each server creates snapshots of compacted logs to guarantee that their logs don't exceed a threshold size. If a follower lags behind, the leader sends RPCs to update the snapshots of its followers.