John Adler

Capstone Abstract

Fleshed out Raft Implementation and Distributed Hash Table Redundancy

For the CS department's Distributed Systems class (CS138), I, along with my partner (class of 2015 Anson Rosenthal), implemented the Tapestry distributed hash table, along with the Raft distributed consensus system. My capstone was focused on expanding these two systems via making the CS138 Raft implementation more complete and integrating Raft and Tapestry.

Although the CS138 Raft implementation was complete with regards to the core of the algorithm, there were some elements described in the original paper which were not implemented as part of the project. One such element was dynamic cluster membership, which I worked on implementing. If a new Raft node wishes to join a currently existing cluster, then it will ask the current leader (to which it will be pointed beforehand) if it can act as an "auditor", a non-voting, non-participating follower. The leader will commit the presence of the auditor to the cluster followers, who will then be able to recognize the auditor's existence in the event of a leadership change. The auditor then receives heartbeats and, by extension, log updates from the leader. This will continue until the auditor is caught up with the leader, at which point it will request to become a fully-fledged part of the cluster. This request, once committed, results in the cluster adding the new node to their configurations, thus allowing any new leaders to know about the existence of the new node. Any new node addition is added to the raft log, meaning that all nodes that have processed a fully up to date log will have added all new nodes in the configuration to their configurations, and thus should be fully up to date.

In addition, I worked on integrating Raft and Tapestry with the goal of making Tapestry more redundant and tolerant of failure. This was accomplished by using Raft as the means by which Tapestry stores it's "blobs" of data. This required making some adjustments to the CS138 Raft implementation, which was previously used only for storing for only storing AGUID, VGUID pairs for file storage, such that it could handle the storage of generic data. The Tapestry "blobstore" itself was then modified such that it look up all blobs in the associated raft cluster. The presence of the Raft cluster allowed the Tapestry node some degree of recovery. If a Tapestry is brought down for whatever reason, it can now be brought back up and have all of its data still available, as Raft provides a means of saving and restoring the state of the Tapestry's blobstore, meaning the Tapestry node can immediately republish blobs upon coming back online.