Cody Ma
Capstone Abstract

Raft Membership Changes

Raft is a protocol that allows nodes in a distributed system to maintain a consensus by keeping a log that is replicated to every node in the cluster. Prior to Raft, the most commonly used algorithm for maintaining distributed consensus was Paxos, but its complexity made it extremely difficult to implement or even understand. Raft's appeal comes from its comparative simplicity.

Raft works by electing a leader in the cluster of nodes, which is responsible for servicing all requests to modify the state machine. When the leader receives such a request, it will alert all other nodes of this request, and when a majority of nodes in the cluster respond to the leader, it will apply the request to the state machine.

A typical Raft cluster will have 5 nodes. As long as a majority of the nodes in the Raft cluster are functional it can continue to operate, which means that a cluster of 5 nodes can tolerate 2 nodes failing. If, however, a server fails and never comes back online, it may be necessary to remove the failing node(s) and add new ones. My capstone project made it possible to change the nodes in a Raft cluster by either adding or removing nodes.

One possible solution to this might have been to take the cluster offline when a membership change was desired, edit a configuration file, then bring the cluster back online. This is undesirable, however, because the cluster is inaccessible when it is offline, and requiring someone to modify the configuration file is prone to human error. Automating this process without taking the cluster offline can be divided into two steps.

Every node in the raft cluster keeps track of the current configuration, namely, the addresses of other nodes in the cluster, as well as the total number of nodes. When the leader receives a request to either add or remove a node, it will switch the cluster to a state known as *joint consensus*. During *joint consensus*, the leader must hear back from a majority of nodes in the current configuration, as well as a majority of nodes in the new configuration before it can apply any changes to the state machine. It is also during this time that the leader alerts the rest of the nodes in the cluster that the number of nodes in the cluster will be changing. Once a majority of the nodes in the old and new configuration tell the leader they are ready to switch to the new one, the leader will send out a message to the nodes that it is now operating under the new configuration, and conclude the transition.