Paper Review: Zookeeper

Yibo Xu

This paper introduces a service for coordinating processes of distributed applications, which aims to provide a simple and high performance kernel for building more complex coordination primitives at the client. It incorporates elements from group messaging, shared registers, and distributed lock services in a replicated, centralized service. The ZooKeeper interface enables a high-performance service implementation.

Large-scale distributed applications require different forms of coordination. Configuration is one of the most basic forms of coordination. When the authors are designing the coordination service, we moved away from implementing specific primitives on the server side, and instead we opted for exposing an API that enables application developers to implement their own primitives. In such way, the authors are able to enable new primitives without requiring changes to the service core.

The ZooKeeper service comprises an ensemble of servers that use replication to achieve high availability and performance. Its high performance enables applications comprising a large number of processes to use such a coordination kernel to manage all aspects of coordination. Caching data on the client side is an important technique to increase the performance of reads.

ZooKeeper provides to its clients the abstraction of a set of data nodes (znodes), organized according to a hierarchical name space. The znodes in this hierarchy are data objects that clients manipulate through the ZooKeeper API. Upon receiving a request, a server prepares it for execution (request processor). If such a request requires coordination among the servers (write requests), then they use an agreement protocol (an implementation of atomic broadcast), and finally servers commit changes to the ZooKeeper database fully replicated across all servers of the ensemble.