ZooKeeper: Wait-free coordination for Internet-scale systems

ZooKeeper is a service for coordinating processes of distributed applications. ZooKeeper provides high performance kernel for building complex coordination primitives at the client. The ZooKeeper interface enables a high-performance service implementation. In addition to the wait-free property, ZooKeeper provides a per client guarantee of FIFO execution of requests and linearizability for all requests that change the ZooKeeper state. These design decisions enable the implementation of a high performance processing pipeline with read requests being satisfied by local servers. ZooKeeper can handle tens to hundreds of thousands of transactions per second. This performance allows ZooKeeper to be used extensively by client applications. ZooKeeper offers a wait-free coordination service with relaxed consistency guarantees for use in distributed systems and building higher level coordination primitives, even blocking and strongly consistent primitives that are often used in distributed applications.

Clients submit requests to ZooKeeper through a client API using a ZooKeeper client library. In addition to exposing the ZooKeeper service interface through the client API, the client library also manages the network connections between the client and ZooKeeper servers. ZooKeeper uses hierarchical namespaces. It provides the abstraction of a set of data nodes (znodes) to the clients. Znodes are referred using standar UNIX notation for file system path. Clients can create either regular znode or an ephemeral znode. ZooKeeper implements watches to allow clients to receive timely notifications of changes without requiring polling. When a client issues a read operation with a watch flag set, the operation completes as normal except that the server promises to notify the client when the information returned has changed. Watches are one-time triggers associated with a session; they are unregistered once triggered or the session closes. Unlike files in file systems, znodes are not designed for general data storage. Instead, znodes map to abstractions of the client application, typically corresponding to meta-data used for coordination purposes.

ZooKeeper guarantees two basic ordering. First, it guarantees linearizable writes i.e all requests that update the state of ZooKeeper are serializable and respect precedence. And second, FIFO client order i.e all requests from a given client are executed in the order that they were sent by the client. Even though ZooKeeper is wait-free, efficient blocking primitives can be implement with ZooKeeper. ZooKeeper's ordering guarantees allow efficient reasoning about system state, and watches allow for efficient waiting.

ZooKeeper provides high availability by replicating the ZooKeeper data on each server that composes the service. 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 Zoo- Keeper database fully replicated across all servers of the ensemble. In the case of read requests, a server simply reads the state of the local database and generates a response to the request. At the heart of the ZooKeepers architecture is: Request Processor – which guarantees that local replicas never diverge, Atomic Broadcast i.e all request that update ZooKeeper state are forwarded to leader, replicated databases and client server interaction.

To conclude, ZooKeeper takes a wait-free approach to the problem of coordinating processes in distributed systems, by exposing wait-free objects to clients. ZooKeeper achieves throughput values of hundreds of thousands of operations per second for read-dominant workloads by using fast reads with watches, both of which served by local replicas.