Zookeeper: Wait-free coordination for Internet scale systems.

Kartikeya Upasani (kuu2101)

Paper Review, 7 November 2016

1 Motivation

Common coordination tasks between processes running on different servers are required by several distributed applications. This is especially required in architectures with a single or no master server.

2 Goal

Developing an API that provides application developers to create and use distributed hierarchical primitives for coordination of processes.

3 Key Idea

Organizing processes in a filesystem-like tree structure, exposing distributed primitives that do not rely on locking.

4 Approach

At its core, Zookeeper is a distributed file system that organizes information about processes in a tree. Ephemeral nodes are special nodes that are removed if the creator disconnects, while sequential nodes are numbered in monotonically increasing order. These two special nodes are utilized for coordination activities such as leader election and locking. Nodes also support watches or event callbacks that can be utilized to respond to updates to a node. Since this is a read-oriented application, the writes are assumed to be infrequent. Hence the writes are full-writes. Specification of data version, and FIFO ordering of reads and writes is used to ensure data consistency.

5 Results

Zookeeper's performance grows as the read:write ratio of test cases grows.

6 Conclusion

Zookeeper provides a simple interface for distributed coordination. Its high throughput allows applications can make extensive use of it, and not only course-grained locking.

7 Comments

Zookeeper is about correctness and consistency of configuration more than throughput. It is thus suitable chiefly for coordination tasks. Though it is a message queue and distributed file system, it cannot be used as them for other applications.