CCIT Report #895 January 2016

Modular Composition of Coordination Services

Kfir Lev-Ari[1], Edward Bortnikov[2], Idit Keidar[1,2], and Alexander Shraer[3]

[1]Viterbi Department of Electrical Engineering, Technion, Haifa, Israel
[2]Yahoo Research, Haifa, Israel
[3]Google, Mountain View, CA, USA

Abstract

Coordination services like ZooKeeper, etcd, Doozer, and Consul are increasingly used by distributed applications for consistent, reliable, and high-speed coordination. When applications execute in multiple geographic regions, coordination service deployments trade-off between performance, (achieved by using independent services in separate regions), and consistency.

We present a system design for modular composition of services that addresses this trade-off. We implement ZooNet, a prototype of this concept over ZooKeeper. ZooNet allows users to compose multiple instances of the service in a consistent fashion, facilitating applications that execute in multiple regions. In ZooNet, clients that access only local data suffer no performance penalty compared to working with a standard single ZooKeeper. Clients that use remote and local ZooKeepers show up to 7x performance improvement compared to consistent solutions available today.

1 Introduction

Many applications nowadays rely on coordination services such as ZooKeeper [28], etcd [9], Chubby [24], Doozer [8], and Consul [5]. A coordination service facilitates maintaining shared state in a consistent and fault-tolerant manner. Such services are commonly used for inter-process coordination (e.g., global locks and leader election), service discovery, configuration and metadata storage, and more.

When applications span multiple data centers, one is faced with a choice between sacrificing performance, as occurs in a cross data center deployment, and forgoing consistency by running coordination services independently in the different data centers. For many applications, the need for consistency outweighs its cost. For example, Akamai [40] and Facebook [41] use strongly-consistent globally distributed coordination services (Facebook's Zeus is an enhanced version of ZooKeeper) for storing configuration files; dependencies among configuration files mandate that multiple users reading such files get consistent versions in order for the system to operate properly. Other examples include global service discovery [4], storage of access-control lists [1] and more.

In this work we leverage the observation that, nevertheless, such workloads tend to be highly partitionable. For example, configuration files of user or email accounts for users in Asia will rarely be accessed outside Asia. Yet currently, systems that wish to ensure consistency in the rare cases of remote access, (like [40, 41]), globally serialize all updates, requiring multiple cross data center messages.

To understand the challenge in providing consistency with less coordination, consider the architecture and semantics of an individual coordination service. Each coordination service is typically replicated for high-availability, and clients submit requests to one of the replicas. Usually, update requests are serialized via a quorum-based protocol such as Paxos [32], Zab [29] or Raft [37]. Reads are served locally by any of the replicas and hence can be somewhat stale but nevertheless represent a valid snapshot. This design entails the typical semantics of coordination services [5, 9, 28] – atomic (linearizable [27]) updates and sequentially-consistent [31] reads. Although such weaker read semantics enable fast local reads, this property makes coordination services non-composable: correct coordination services may fail to provide consistency when combined. In other words, a workload accessing *multiple* consistent coordination services may not be consistent, as we illustrate in Section 2. This shifts the burden of providing consistency back to the application, beating the purpose of using coordination services in the first place.

In Section 3 we present a system design for modular composition of coordination services, which addresses this challenge. We propose deploying a single coor-