CATS: A Linearizable and Self-Organizing Key-Value Store

Cosmin Arad  Tallat M. Shafaat  Seif Haridi

Distributed key-value stores provide scalable, fault-tolerant, and self-organizing storage services, but fall short of guaranteeing linearizable consistency in partially synchronous, lossy, partitionable, and dynamic networks, when data is distributed and replicated automatically by the principle of consistent hashing [14]. This work introduces consistent quorums as a solution for achieving atomic consistency. We present the design and implementation of CATS, a key-value store which uses consistent quorums to guarantee linearizability and partition tolerance in such adverse and dynamic network conditions. CATS is scalable, elastic, and self-organizing; key properties for modern cloud storage middleware. Our system evaluation shows that consistency can be achieved with practical performance and modest overhead: 5% decrease in throughput for read-intensive workloads, and 25% throughput loss for write-intensive workloads. CATS delivers submillisecond operation latencies under light load, single-digit millisecond operation latencies at 50% load, and it sustains a throughput of one thousand operations per second, per server, while scaling linearly to hundreds of servers.

Distributed key-value stores, such as Cassandra [15] and Dynamo [8], employ principles from DHTs [21] to build scalable and self-managing data stores. In contrast to CATS, these systems chose availability over atomic consistency, hence only providing eventual consistency [22]. While eventual consistency is sufficient for some applications, the complexities of merging divergent replicas can be non-trivial. We avoid the complexities entailed by eventual consistency while providing scalable storage for critical applications which need atomic consistency, guaranteeing it at the cost of a modest decrease in throughput.

To handle dynamic networks, atomic registers were extended by protocols such as RAMBO [17], RAMBO II [9], RDS [7] and DynaStore [1] to be reconfigurable. Similarly, SMART [16] enabled reconfiguration in replicated state machines. With consistent quorums we provide high-throughput read/write operations without paying the full cost of state machine replication which needs coordination for every operation. Moreover, our design does not depend on electing a single leader and the complexities that come with that [5]. While these systems can handle dynamism and provide atomic consistency, they are not scalable as they were not designed to partition the data across a large number of machines. The novelty of CATS is in extending the reconfiguration techniques contributed by these works, such that they can be used at large scale, in order to build a system that is completely decentralized and self-managing.

Distributed coordination systems such as Chubby [4] and ZooKeeper [11, 13], provide linearizability and crash-recovery, but are not scalable. Master-based key-value stores, such as Bigtable [6], HBase [12], and MongoDB [18], rely on a central server for coordination and data partitioning. Similarly, Spinnaker [19] uses Zookeeper [11]. Since these systems are centralized, their scalability is limited. In contrast, CATS is decentralized and all nodes are symmetric, allowing for unlimited scalability.

Similar to CATS, Scatter [10] is a scalable and consistent key-value store. Scatter employs an extra subsystem and policies to decide when to reconfigure replication groups. In contrast, CATS has a simpler and more efficient reconfiguration protocol – both in the number of messages and message delays – which does not require distributed transactions. We focus on consistent-hashing at the node level, which makes our approach directly implementable in existing key-value stores like Cassandra [15]. The main distinguishing advantage of CATS over Scatter is CATS’ ability to handle network partitions and mergers, an aspect largely ignored in Scatter. Once network partitions cease, CATS merges partitioned subsystems into a single overlay, while Scatter will continue to operate as separate overlays. Where Scatter provides scalability and consistency, CATS provides scalability, consistency, and partition tolerance.

The design, implementation, and evaluation of CATS are available in a separate technical report [3, 2, 20].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s).
References


