Stronger Semantics for Low-Latency Geo-Replicated Storage

Wyatt Lloyd[⋆], Michael J. Freedman[◇], Michael Kaminsky[†], and David G. Andersen[‡]
[⋆◇]Princeton University, [†]Intel Labs, [‡]Carnegie Mellon University
[⋆]wlloyd@cs.princeton.edu, [◇]mfreed@cs.princeton.edu, [†]michael.e.kaminsky@intel.com, [‡]dga@cmu.edu
[⋆]student, no demo planned

1 Extended Abstract

Large-scale data stores are a critical infrastructure component of many Internet services. In this work, we address the problem of building a geo-replicated data store targeted at applications that demand fast response times. Such applications are now common: Amazon, EBay, and Google all claim that a slight increase in user-perceived latency translates into concrete revenue loss

Providing *low latency* to the end-user requires two properties from the underlying storage system. First, storage nodes must be near the user to avoid long-distance round trip times; thus, data must be replicated geographically to handle users from diverse locations. Second, the storage layer itself must be fast: client reads and writes must be local to that nearby datacenter and not traverse the wide area. Geo-replicated storage also provides the important benefits of availability and fault tolerance.

Beyond low latency, many services benefit from a *rich data model*. Key-value storage—perhaps the simplest data model provided by data stores—is used by a number of services today The simplicity of this data model, however, makes building a number of interesting services overly arduous, particularly compared to the column-family data models offered by systems like BigTable nd Cassandra These rich data models provide hierarchical sorted column-families and numerical counters. Column-families are well-matched to services such as Facebook, while counter columns are particularly useful for numerical statistics, as used by collaborative filtering (Digg, Reddit), likes (Facebook), or re-tweets (Twitter).

Unfortunately, to our knowledge, no existing geo-replicated data store provides guaranteed low latency, a rich column-family data model, and *stronger consistency semantics*: consistency guarantees stronger than the weakest choice—eventual consistency—and support for atomic updates and transactions. This work presents Eiger a system that achieves all three properties.

The consistency model Eiger provides is tempered by impossibility results: the strongest forms of consistency—such as linearizability, sequential, and serializability—are impossible to achieve with low latency that is, latency less than the network delay between datacenters). Yet, some forms of stronger-than-eventual consistency are still possible and useful, e.g., *causal consistency* and they can benefit system developers and users. In addition, *read-only* and *write-only transactions* that execute a batch of read or write operations at the same logical time can strengthen the semantics provided to a programmer.

A key challenge of this work is to meet these three goals while *scaling* to a large numbers of nodes in a single datacenter, which acts as a single logical replica. Traditional solutions in this space such as Bayou assume a single node per replica and rely on techniques such as log exchange to provide consistency. Log exchange, however, requires serialization through a single node, which does not scale to multi-node replicas.

Our Eiger system is a scalable geo-replicated data store that achieves our three goals. Like COPS, Eiger tracks dependencies to ensure consistency; instead of COPS' dependencies on versions of keys, however, Eiger tracks dependencies on operations. Yet, its mechanisms do not simply harken back to the transaction logs common to databases. Unlike those logs, Eiger's operations may depend on those executed on other nodes, and an operation may correspond to a transaction that involves keys stored on different nodes.

The contributions of this work are as follows:

- The design of a low-latency, causally-consistent data store based on a column-family data model, including all the intricacies necessary to offer abstractions such as column families and counter columns.
- A novel non-blocking read-only transaction algorithm that is both performant and partition tolerant.
- A novel write-only transaction algorithm that atomically writes a set of keys, is lock-free (low latency), and does not block concurrent read transactions.
- An evaluation that shows Eiger has performance competitive to eventually-consistent Cassandra.