CCIT Report #827 March 2013

ACID-RAIN: ACID Transactions in a Resilient Archive with Independent Nodes

Ittay Eyal[1], Ken Birman[2], Idit Keidar[1], and Robbert van-Renesse[2]

[1]Department of Electrical Engineering, Technion, Haifa, Israel
[2]Department of Computer Science, Cornell Univerisity, Ithaca, NY, USA

Abstract

In cloud-scale data centers, it is common to shard data across many nodes, each maintaining a small subset of the data. Although ACID transactions are desirable, architects often avoid them due to performance concerns. We present a novel architecture for support of low-latency high-throughput ACID transactions in a Resilient Archive with Independent Nodes (ACID-RAIN). ACID-RAIN uses logs in a novel way, limiting reliability to a single scalable tier. A large set of independent fault-prone nodes form an outer layer that caches the sharded data, backed by a set of independent highly available log services. ACID-RAIN dramatically reduces concurrency conflicts by using prediction to order transactions before they take actions that would lead to an abort. Simulations using the Transactional-YCSB workloads demonstrate scalability and effective contention handling.

1 Introduction

Large-scale data-center computing systems often employ massive data sets, *sharded* (partitioned) over many storage nodes. When client transactions access shared data items, the issue of consistency arises. Ideally, we would use a system with ACID transactions [2, 18, 1], because this model facilitates reasoning about system properties and makes possible a variety of high-assurance guarantees. Nonetheless, the ACID model is widely avoided due to efficiency concerns [15].

In case ACID transactions are needed, one of two approach is commonly used. The first is to use a central highly available certification entity (e.g., [16, 23]) for serializing transactions. However, such a central certification entity has limited throughput, and therefore, this approach cannot scale beyond a certain point.

Another option is to use a combination of locking or optimistic concurrency control (OCC) with timestamped version management in each shard, together with two phase commit (2PC) across shards. However, 2PC is generally avoided in high-availability systems due to performance and fault-tolerance concerns. In case of failures, specifically, of the 2PC coordinator, which is not a rare event in a large-scale system, all potentially conflicting transactions must block until the failure is mended. To avoid that, existing systems [6, 9] replace the 2PC coordinator with a highly available one. This severely harms throughput, as we demonstrate in Section 5.

In this paper, we present ACID-RAIN — an architecture for ACID transactions in a Resilient Archive with Independent Nodes. It is depicted in Figure 1. Our approach uses logs in a novel manner. A set of independent highly-available logs collaboratively describe the state of the entire system, i.e., one would need to combine all logs in order to learn the global state. Each log is accessed through an *Object Manager* (*OM*) that caches the data and provides the data structure abstraction. *Transaction Managers* (*TM*s) provide the atomic transaction abstraction, certifying a given transaction by checking for conflicts in each log via its OM. The benefit of our approach is that other than the logs, no system entities are required to be highly-available. OMs and TMs that are suspected to have failed can be instantly replaced; safety is not violated by multiple copies running concurrently.

Our system uses a form of OCC: OMs respond to concurrent TM instructions with no locks. To improve latency, the OMs serve requests from speculative local data structures, referring to the logs only for certification. To decrease abort rate, we use *predictors* that foresee the likely access pattern of transactions; such predictors can be implemented with machine learning tools [25]. To leverage prediction, a transaction *leases* a version of an object for its use. Note that unlike locks, failure to respect a lease does not violate safety, and therefore does not delay OM restoration on failures.

We evaluate ACID-RAIN through simulation with the transactional YCSB benchmark [13, 14]. We demonstrate the algorithm's linear scalability, compared to the other approaches mentioned above, and the effectiveness of using accurate and inaccurate predictors.