JEPSEN

Distributed Systems Safety Research

About Jepsen

Jepsen is an effort to improve the safety of distributed databases, queues, consensus systems, etc. We maintain an open source software library for systems testing, as well as blog posts and conference talks exploring particular systems' failure modes. In each analysis we explore whether the system lives up to its documentation's claims, file new bugs, and suggest recommendations for operators.

Recent Work

- Jepsen and the etcd team worked to verify etcd's key-value operations, locks, and watches. We found no problems with key-value operations or watches: kv operations, including multi-key transactions, appeared to be strict serializable, and watches delivered every change to keys in order, with the exception of an undocumented edge case around revision zero. Locks, however, were fundamentally unsafe: they did not, and can not, guarantee mutual exclusion

Jepsen pushes vendors to make accurate claims and test their software rigorously, helps users choose databases and queues that fit their needs, and teaches engineers how to evaluate distributed systems correctness for themselves.

In addition to public analyses, Jepsen offers technical talks, training classes, and distributed systems consulting services.

Other Resources

- Github
- Twitter
- Announcements mailing list
- General discussion mailing list

in asynchronous networks. This theoretical risk was exacerbated by an implementation bug—now fixed in master. The etcd team is preparing new documentation around safety guarantees.

- We collaborated with YugaByte to validate YugaByte DB at version 1.3.1's beta support for serializable SQL transactions. We found two safety issues: a race condition allowing `DEFAULT` columns to be initialized to `NULL`, and a serializability violation (G2-item) associated with tablet servers losing their connection to master nodes. We also found several problems with DDL statements like table creation, and some availability issues, such as slow recovery during network partitions, and process leaks. YugaByte has addressed the G2 issue as well as some of the other bugs, and is working on remaining issues.

- Together with PingCAP, we tested TiDB 2.1.7 through 3.0.0-rc.2, and found that by default, TiDB violated snapshot isolation, allowing read skew, lost updates, and incompatible orders, thanks to two separate auto-retry mechanisms which blindly re-applied conflicting writes. 3.0.0-rc.2 changed the default settings to provide snapshot isolation by default, as well as fixing a race condition in table creation, and some crashes on startup.

- In cooperation with YugaByte, we identified three safety issues in YugaByte DB, including read skew and data corruption under normal operating conditions, read skew and data corruption during clock skew, and, under rare circumstances involving multiple network partitions, the loss of acknowledged inserts. YugaByte DB

addressed all three of these problems, as well as several other issues, in 1.2.0.

- Fauna and Jepsen worked together to identify 19 issues in FaunaDB, a distributed transactional document store based on the Calvin protocol. In addition to minor problems with schema changes and availability, we found that indices could fail to return negative integer values, skip records at the end of pages, and exhibit read skew. Temporal queries could observe older states than the timestamp they requested, and exhibited occasional read skew. We also found nonmonotonic reads, long fork, and read skew in development releases. We found that basic key-value operations in FaunaDB 2.5.4 appeared to provide snapshot isolation up to strict serializability, depending on workload. FaunaDB has addressed every serious issue we identified in version 2.6.0, and fixes for minor issues, like concurrent schema modification, are slated for upcoming releases.

- We worked with MongoDB to explore sharded single-document consistency and MongoDB's new support for causally consistent sessions. We found that sharded clusters appeared to preserve single-document linearizability and did not lose inserted documents during shard rebalancing and network partitions. However, causal sessions required majority reads and writes; at the default consistency levels, causal sessions did not preserve claimed ordering invariants. MongoDB has updated their documentation to reflect this.

- We analyzed Dgraph, a distributed graph database, and identified numerous deadlocks, crashes, and consistency violations, including the loss and

corruption of records, even in healthy clusters. Dgraph addressed many of these issues during our collaboration, and continues to work on remaining problems.

- Together with Aerospike, we validated their next-generation consensus system, confirming two known data-loss scenarios due to process pauses and crashes, and discovering a previously unknown bug in their internal RPC proxy mechanism which allowed clients to see successfully applied updates as definite failures. Aerospike fixed this bug, added an option to require nodes write to disk before acknowledging operations to clients, and plans to extend the maximum clock skew their consensus system can tolerate.