Technical Perspective: FoundationDB: A Distributed Unbundled Transactional Key Value Store

Alfons Kemper
Technical University of Munich

With the emergence of (geographically) distributed data mangement in cloud infrastructures the key value systems were promoted as so-called NoSQL systems. In order to achieve maximum availability and performance these KV stores sacrificed the "holy grail" of database consistency and relied on relaxed consistency models, such as *eventual consistency*. This was a consequence of Eric Brewer's so-called CAP observation (aka Theorem) [1] stating that only two of the three desiderata of distributed systems could possibly be satisfied at the same time: (1) Consistency, where every read operation receives the most recent committed write, (2) Availability which nowadays typically strives for the so-called many (e.g., seven or even nine) nines meaning 99.99999% up-time, (3) Partition tolerance which demands that the system has to remain operational even under severe network malfunctions. As a consequnce, NoSQL system designers traded consistency for higher availability and network fault tolerance. The relaxed replica convergence models were categorized by Werner Vogels [2] of Amazon in 2009. However, many mission critical applications even in cloud settings do require stronger consistency guarantees. Eventual consistency only ensures convergence to the same state of all replicas. FoundationDB, on the other hand, is a scalable distributed key value store with strong consistency guarantees. It started 10 years ago as an open source project and is now widely used as a mission-critical backbone repository in cloud infrastructures, such as Apple and Snowflake. In this respect FoundationDB re-unites the *NoSQL* paradigm of high availability and low latency with the ACID (Atomicity, Consistency, Isolation, Durability) guarantees imposed by traditional database systems. This new breed of systems is therefore coined *NewSQL* – albeit not all offering SQL interfaces. For scalability and elasticity in cloud infrastructures FoundationDB exhibits a fully disaggregated architecture consisting of a storage system (SS), a logging system (LS), and a separated transaction system (TS). Storage Servers are decoupled from Log Servers, which maintain the "ground truth". Storage Servers continuously apply log records from Log Servers in order to lag only slightly behind the transaction commit state – in production system measurements the authors report a lag in the order of just milliseconds only. The transaction system employs an optimistic multi version concurrency control (MVCC) scheme with a subsequent verification phase to ensure strict serializability. Thereby, read only transactions are not penalized as they access versions that were committed by the time they started. For this purpose a so-called Sequencer process determines the corresponding time stamp that is then observed by proxies to access the correct version from storage servers. For write transactions, the Sequencer assigns the commit time stamp with which the resolvers verify strict serializability by comparing the transaction's read set (i.e., the key ranges of key value pairs that were accessed in the course of the transaction) with the write sets of transactions committed in the mean time. Successfully verified transactions are made durable by writing their logs to multiple log servers for fault tolerance. These log servers can be replicated across distinct (geographic) regions to tolerate failures (such as power outages) within an entire region. The FoundationDB system can be configured for synchronous as well as asynchronous log transferral – thereby trading off between safety and latency requirements. In case of a primary server failure the synchronous mode guarantees seamless instantaneous takeover by the secondary server – albeit incurring extra latency overhead during commit processing. Under asynchronous mode the primary server maintains several satellites within the same region which, in case of failure, can submit the log's suffix that was not yet sent to the secondary server in a remote region. Thus, the tradeoff between performance/responsiveness and failure resilience is one of the foremost goals of the FoundationDB design. Correctness and Robustness was also a key aspect in the development life cycle of the FoundationDB system. In order to achieve this the team relied on simulation testing that allowed to run the distributed software code in a deterministic manner for error reproducibility and extensive test coverage.

To conclude, the design of FoundationDB lays the "foundation" for future cloud information systems that have to balance performance and consistency requirements.

1. REFERENCES

[1] Armando Fox and Eric Brewer. Harvest, Yield and Scalable Tolerant Systems. In *Proc. 7th Workshop Hot Topics in Operating Systems (HotOS 99)*, IEEE CS, 1999, pg. 174–178.

[2] Werner Vogels. Eventually consistent. *Communications of the ACM*. 52: 40–44, 2009.