rqlite

Replicating SQLite using Raft Consensus

Philip O’Toole
http://www.philipotoole.com

ACM Pittsburgh, October 8th 2018

About Me

- Engineering Manager at Google Pittsburgh.
- Recent transplant from the San Francisco Bay Area.
- Previously a full-time core-developer at InfluxDB, an open-source time-series database written in Go.
- Led various other software teams building networking software, search engines, and distributed systems.

About rqlite

- rqlite is a distributed system, which fully replicates a SQLite database, using the Raft consensus protocol.
- Source available at https://github/com/rqlite/rqlite
- Development began 4 years ago.
  - The first simple working system was available very early on.
- The current release is v4.3.0.

Goal for this talk

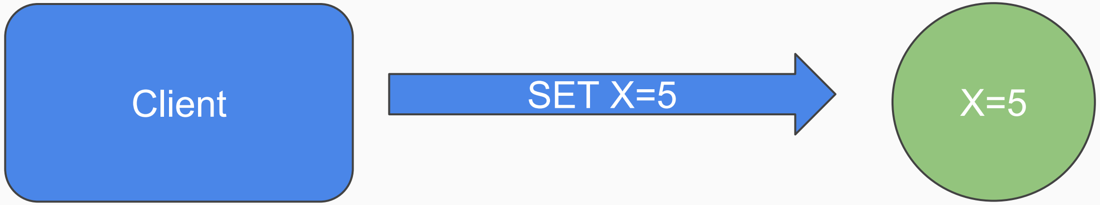
To understand the previous slide!

Why a distributed system?

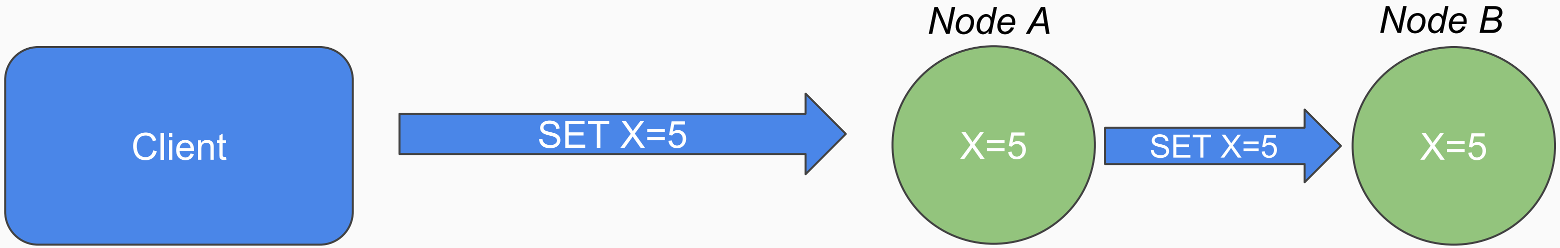
- A distributed system provides reliability
  - Your data is located in multiple places
  - The computation is available from multiple places.
- A distributed system often -- but not always -- also provides scalability.
  - Distributed systems may be more powerful.

What is the challenge?

It’s easy to set the value of a single node

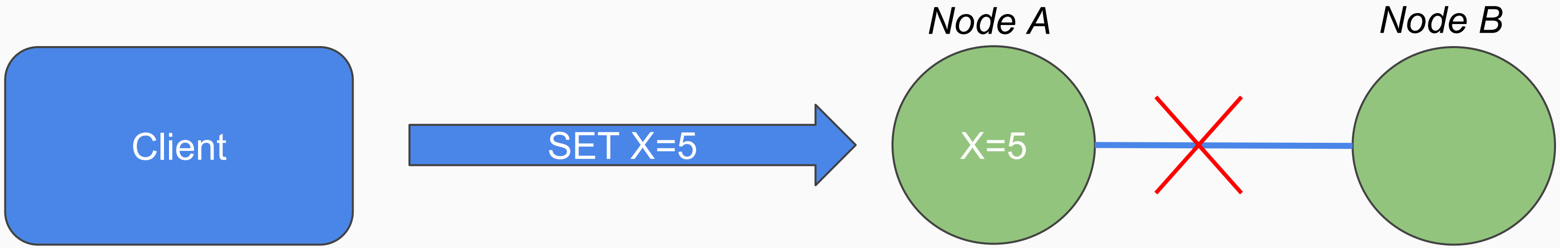


What if the value should be replicated?



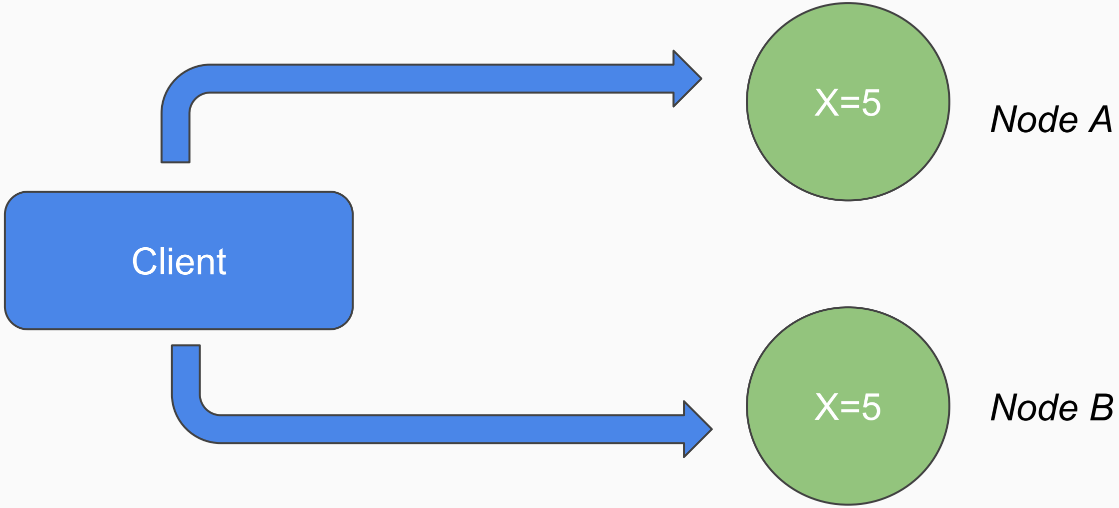
Multi-node system - receiving node replicates data

What if the nodes can’t communicate?



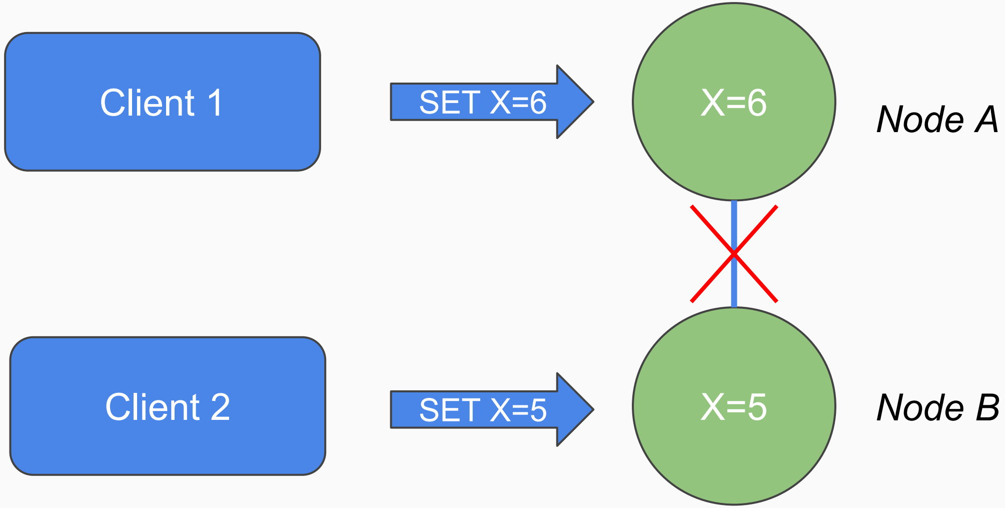
This failure is known as a *partition*.

What if the value should be replicated?



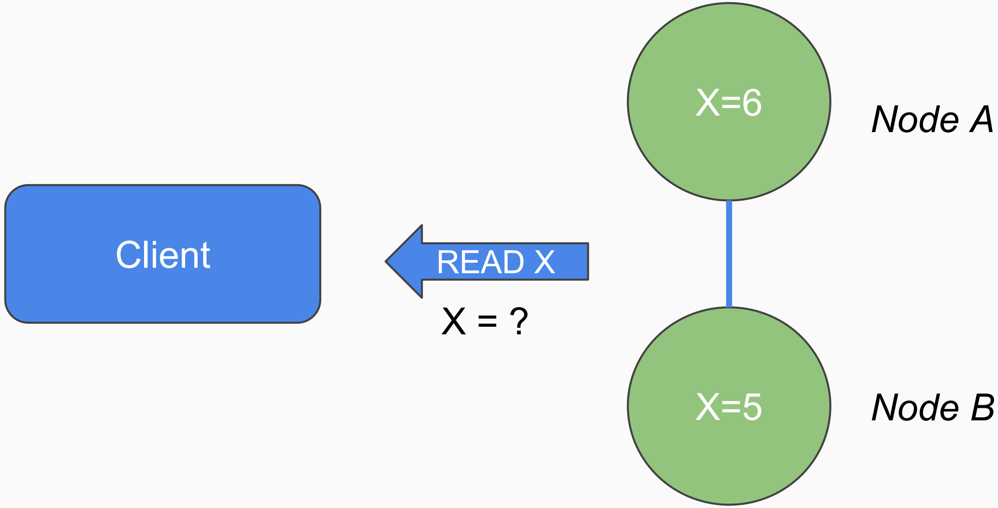
Client replication: multi-node system - each node must support changing state

What if the nodes can’t communicate?



Client replication: multi-node system - each node must support changing state

Communication restored



What value should be read for the node?

This problem is known as *Distributed Consensus*

What is the solution?

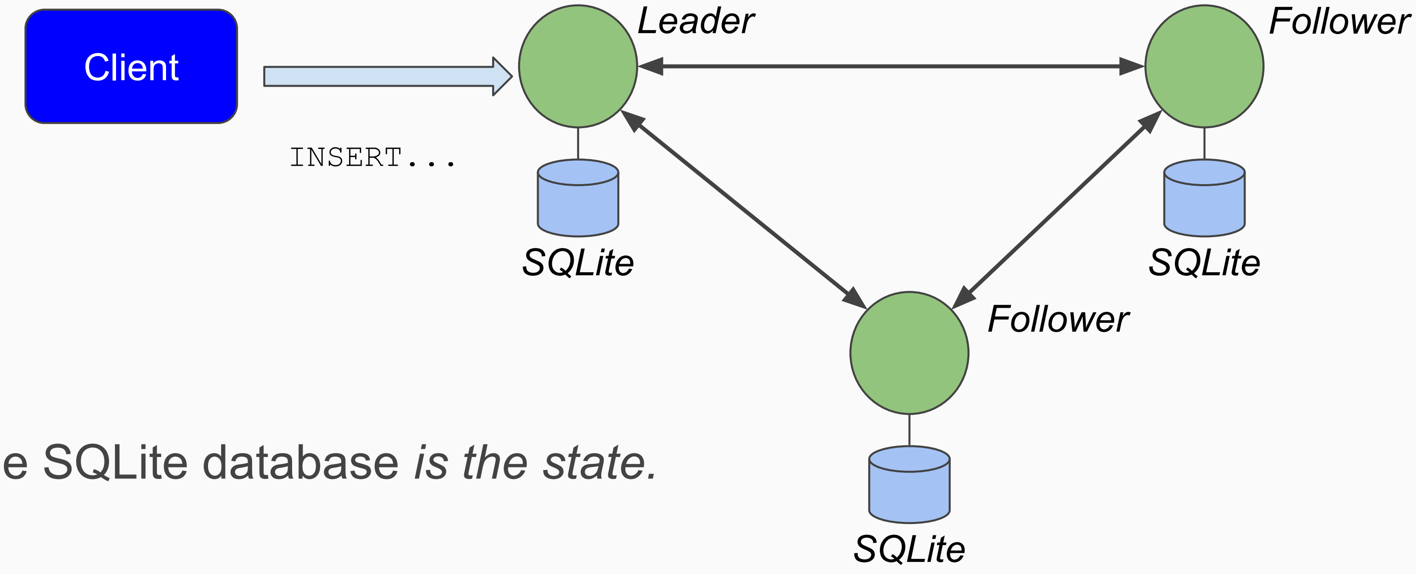
- Paxos
  - Famously difficult to understand
- ZAB (Zookeeper Atomic Broadcast)
  - Created by Yahoo! Research
- Raft
  - Diego Ongaro and John Ousterhout at Stanford.
  - We are going to focus on this technique.

Secret Lives of Data: Raft

Why replicate SQLite?

- Rock-solid relational database, contained within a single C-source file.
- With replication, you get reliability.
- Easy installation and deployment, thanks to Go.
  - Go compiles to single, statically-linked binary.
- Lightweight operation.

rqlite cluster

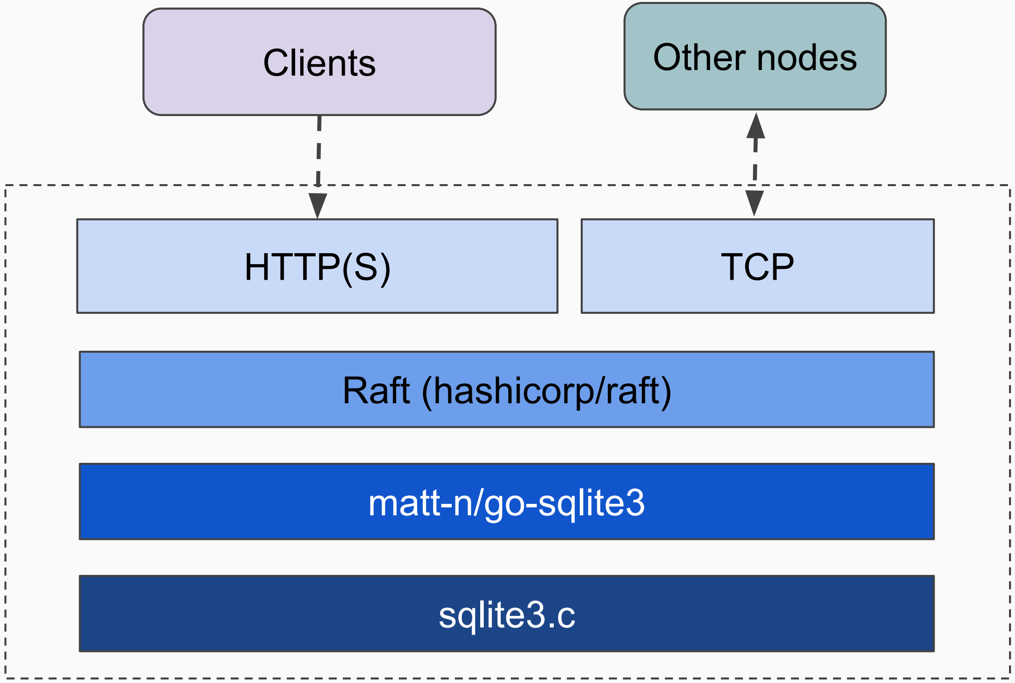


The SQLite database *is the state.*

rqlite cluster

- Clusters are most practical when 3, 5, or 7 nodes in size.
  - Even numbers don’t get you anything.
    - A 3-node cluster has a quorum of 2, which means it can tolerate loss of a single node.
    - A 4-node cluster has quorum of 3, which it too can only tolerate loss of a single node.
    - You must go to a 5-node cluster to tolerate loss of 2 nodes.
- Not suitable for replicating massive clusters such as all the SQLite databases on a smartphone network.

rqlite node design



Hashicorp Raft

- Raft implementation in Go from Hashicorp.
  - Available at https://github.com/hashicorp/raft
- Used by Consul, Nomad, and InfluxDB.
- Another well-known Go implementation of Raft also exists.
  - Written by CoreOS.
  - Forms the basis of etcd, which in turn is critical to Kubernetes.

Integrating with Hashicorp Raft

- Simply implement five functions

```
Apply(l *raft.Log) interface{} // Apply a committed entry to the state machine

Snapshot() (raft.FSMSnapshot, error) // Returns a snapshot of the state machine

Restore(rc io.ReadCloser) error // Create state machine from snapshot

Persist(sink raft.SnapshotSink) // Write snapshot to persistent storage

Release()                       // Snapshot release. Usually a no-op
```

Lessons learned

- “Plan to throw one away. You will, anyhow” -- Fred Brooks.
- Early versions of rqlite suffered from a poor API
  - Study existing art.
- Non-idiomatic Go
  - Study a language’s style before starting a significant project.
- Used go-raft for consensus, which was nearing end-of-life.

rqlite v4.3.0

- Idiomatic Go code
- Much better API, thanks to experience with InfluxDB.
- Limited use of 3rd-party libraries.
- Good use of Go interfaces, which allowed for better testing.
- Direct use Go of SQLite package.
  - Skip db/sql abstraction.

What rqlite can do

- With it you’ve got a lightweight and reliable distributed store for relational data.
- You could use rqlite as part of a larger system, as a central store for some critical relational data, without having to run a heavier solution like MySQL.
- rqlite might also be an effective way to provide small number of SQLite read-replicas.

What you can do

With Go and Raft you can build real distributed systems **today**. So give it a try!

References

- https://github.com/rqlite
- https://github.com/hashicorp/raft
- https://github.com/mattn/go-sqlite3
- https://github.com/otoolep/hraftd
- https://raft.github.io
- http://thesecretlivesofdata.com/raft/