Cockroach

A Scalable, Geo-Replicated, Transactional Datastore

What kind of datastore?

- currently a sorted key-value store, but:
- structured and SQL-like layers are coming
- in the end, it should feel like a SQL database (unless you want the lower layers!) with indexes, joins and more
- written in Go

Scalable, Geo-Replicated, Transactional

???

Scalable

Geo-Replicated

Transactional

separates Cockroach from NoSQL:

Consistent & Highly Available is *difficult*:

- apps can do it, but it is very hard (think: encryption)
- the database should do this once, correctly
- the cost is consensus latency
- CockroachDB has transactions that fully deserve the name

```
opts := client.TransactionOptions{Name: "example put"
c.RunTransaction(&opts, func(txn *client.KV) error {
  // serializable context!
  gr := proto.GetResponse{}
  txn.Call(proto.Get,
           proto.GetArgs(proto.Key("key1")), &gr)
  txn.Call(proto.Put,
           proto.PutArgs(proto.Key("key2"),
             append(gr.Value.Bytes, []byte("-new"))),
           &proto.PutResponse{})
  return nil
})
```

Why

History

- SQL “not” scalable or highly available, but transactional
- PostgreSQL, MySQL, Oracle, DB2, . . .
- NoSQL scalable and highly available, but not consistent
- BigTable, Cassandra, . . .
- NewSQL scalable, highly available, transactions
- Spanner, CockroachDB, . . .

History at Google

- 2004: BigTable
- 2006: Megastore (on top of BigTable)
- transactional (but slow and complex)
- 2012: Spanner
- fully linearizable (hence consistent)

> *"We believe it is better to have application programmers deal with performance problems due to overuse of transactions as bottlenecks arise, rather than always coding around the lack of transactions."*

Corbett et al., Spanner paper, 2012

Google Spanner

is basically what you would get if SQL and NoSQL had a designer baby that combined both their advantages: scalable, highly available, transactional

but. . .

- only Google can have it
- hardware: atomic clocks, GPS receivers
- inhomogeneous infrastructure: TrueTime API, Chubby, Collossus, . . .

We want you

- to have something like Spanner
- platform semirelational database
- fault-tolerant, transactional, scalable, fast (enough)
- but simpler than Spanner
- simple homogenous infrastructure
- no hardware requirements
- and OpenSource
- this stuff is hard - trust nobody
- see: Jepsen series

How?

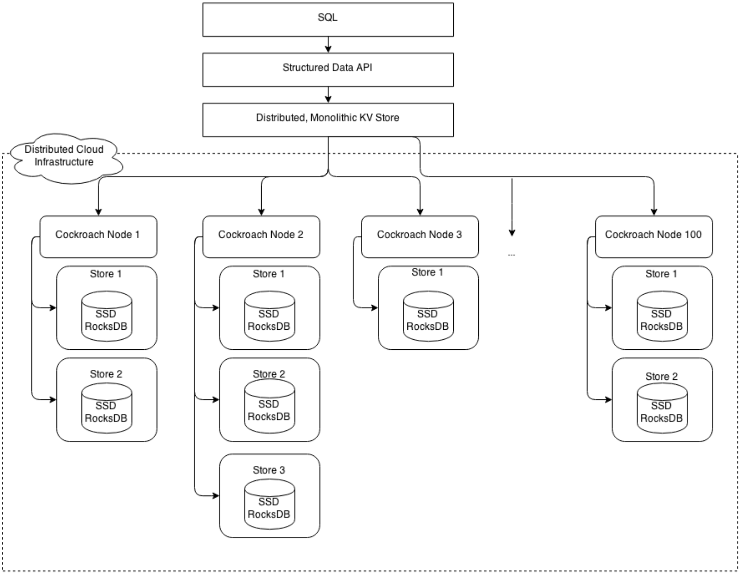

Figure 1: Cockroach Architecture

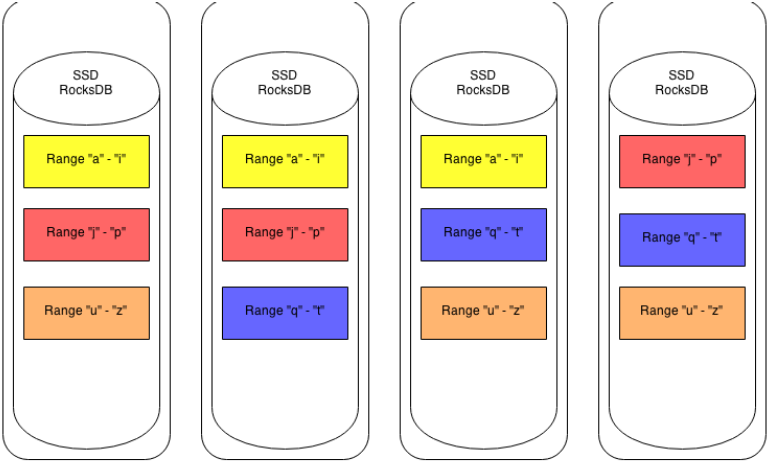

Figure 2: Cockroach Ranges

Distributed Transactions

- lock free
- serializable snapshot isolation semantics
- transactions logically don't overlap
- transaction restarts are expected (and normal)
- linearizability for common cases
- a rare concern in practice
- can enforce for all cases when time signal is good

Under The Hood

- variation of two phase commit
- txn writes stored as MVCC “intents”
- central transaction table:
- single key/txn: status, timestamp, priority, . . .
- modified by concurrent txns - first writer wins
- the single source of truth
- 2nd phase more efficient – 1 write to transaction table entry
- intents resolved after commit - correctness doesn’t need it!

Who?

Summary