111

Scalable

Renlicated

Transaction

Who?

Summary

Cockroach

???

Scalable

Geo-

Transaction

vviiy

vvno

Summary

A Scalable, Geo-Replicated, Transactional Datastore

What kind of datastore?

Cockroach

111

Scalable

Geo-Poplicated

Transaction

Wh

VVho!

- 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-

+ ...

vviio:

Summary

Scalable, Geo-Replicated, Transactional

???

Scalable

Replicated

Transactions

Transaction

Why

Who?

Summary

???

???

Scalable

Replicated

Transactiona

vviio:

777

Scalable

Geo-Replicated

Transactiona

Why

.

Who?

Summary

Scalable

111

calable

Geo-Replicated

Transactiona

Why

Who

Summary

Geo-Replicated

111

Scalable

Replicated

Transactional

M/hv

Who?

Summary

Transactional

separates Cockroach from NoSQL:

Cockroach

111

Scalable

Geo-Replicated

Transactional

M/by

VVho

Summar

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

```
Cockroach
```

???

Scalable

Geo-

Transactional

.

0 0 11

Who

```
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
})
```

111

Scalable

Replicated

Transactional

vvny

Who?

???

Scalable

Renlicated

Transactions

Why

How!

Who?

Summary

Why

Scalable

Geo-

Transaction

Why

.

VVho

- 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

Cockroach

111

Transaction

Why

11000

VVho

Summar

2004: BigTable

2006: Megastore (on top of BigTable)

transactional (but slow and complex)

2012: Spanner

fully linearizable (hence consistent)

:::

Scalable

Geo-

Transaction:

Why

VVIIO

Summary

"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

Cockroach

111

Scalable

Geo-

Transaction

Why

.

Who

Summary

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

111

Scalable

plicated

Transaction

Why

Who

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

???

Scalable

Geo-

Transactiona

Why

.

VVIIO

- 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

111

Scalable

Renlicated

Transactiona

Why

vvno!

???

Scalable

Replicated

Transactiona

How?

M/L = 2

Summary

How?

777

Scalable

Geo-

...

Transactio

Why

How?

Who?

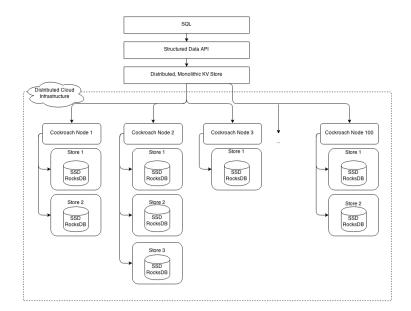


Figure 1: Cockroach Architecture

SSD RocksDB SSD RocksDB SSD RocksDB SSD RocksDB Range "a" - "i" Range "a" - "i" Range "a" - "i" Range "j" - "p" How? Range "j" - "p" Range "j" - "p" Range "q" - "t" Range "q" - "t" Range "u" - "z" Range "q" - "t" Range "u" - "z" Range "u" - "z"

Figure 2: Cockroach Ranges

Distributed Transactions

Cockroach

111

Transactional

Transactiona

How?

A /I. . .

V V 110

- 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

Cockroach

111

Scalable

Geo-

Transaction

Wh

How?

Who'

Summar

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!

???

Scalable

Donlington

.

vvny

How?

Who?

???

Scalable

Renlicated

Transactions

vvny

Who?

Summary

Who?

???

Scalable

Replicated

Transactions

Why

Who?

Summary