CockroachDB 设计与实现

by 刘奇
微博 @goroutine

DTCC

2015中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2015

大数据技术探索和价值发现

IT168.com ChinaUnix ITPUB

数据库的演化

SQL时代：MySql, PostgreSQL

NoSQL: MongoDB, redis, HBase…

NewSQL: Google F1, FoundationDB, CockroachDB

Why

Transaction + Scale

Google的历程

2004: BigTable

eventually consistent NoSQL

2006: Megastore (on top of BigTable)

transactional, slow, complex

2010: remove sharding mysql

2012: Spanner (+F1 on top of it)

semi-relational, fully linearizable

历史上的努力

cobar: just sharding. Simple

vitess: from youtube. Complex

还有无数各大公司造的轮子

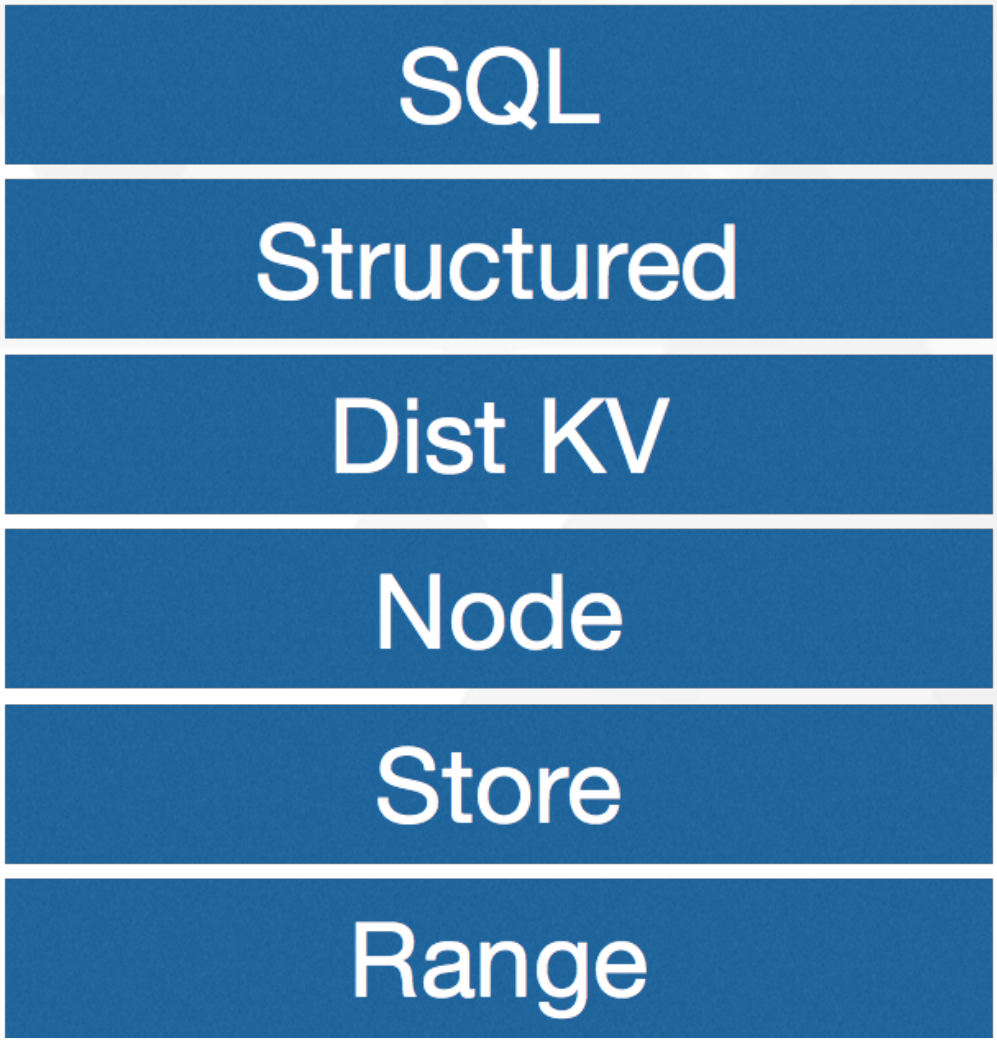
....

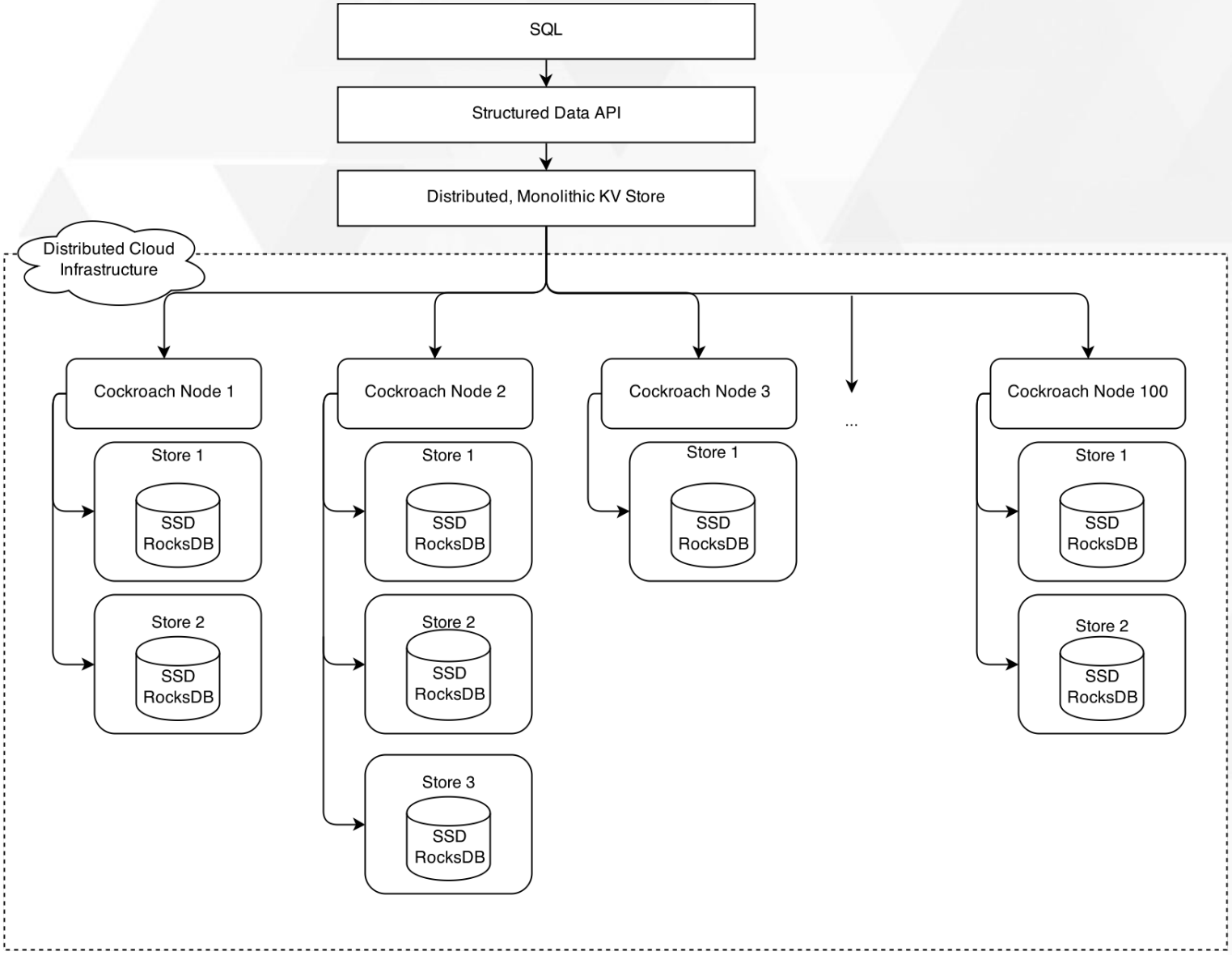
No distributed transaction !!!!!!!

CockroachDB

- Cockroach is CP in CAP
- “A”vailable not same as “H”ighly “A” vailable

Architecture




SQL
Structured Data API
Distributed, Monolithic KV Store
Distributed Cloud Infrastructure
Cockroach Node 1
Store 1
SSD
RocksDB
Store 2
SSD
RocksDB
Cockroach Node 2
Store 1
SSD
RocksDB
Store 2
SSD
RocksDB
Store 3
SSD
RocksDB
Cockroach Node 3
Store 1
SSD
RocksDB
...
Cockroach Node 100
Store 1
SSD
RocksDB
Store 2
SSD
RocksDB

事务原理

- Variation of two phase commit
- Txn writes stored as MVCC “intents”
- Txn table has a single key / txn
  - Stores txn status, timestamp, priority
  - Modified by concurrent txns - first writer wins
  - The single source of truth
- 2nd phase more efficient -- 1 write
- Intents resolved after commit

Txn table在哪里？

- 也作为kv存在某个range里面，具体由txn.Key决定

Linearizability

- Serializable for all cases
- Temporal reverse?
- Client decision: perf / correctness
- Max Timestamp
  - Passed to order causal txns
- Commit wait
  - Always waiting means true linearizability
  - More accurate clocks = less wait

选择读时间戳

- Spanner: TrueTime always reads committed value
- Cockroach:
  - Doesn’t wait on writes
  - Sometimes waits on reads
  - For txn, choose T_{start}, T_{max}
  - If read encounters timestamp in “uncertainty” window, restart read-only txn
  - Shrinking window means max wait is clock skew

Uncertainty

- --------ts----------txn.MaxTs---------->
- 剪头指向的绝对时间方向

关于 存储引擎

- 重用已有的成果, 不是整个系统的重点
- rocksdb 已经足够快了
- 设计上考虑支持多种存储引擎

关于 hlc

- hybird logic clocks
- 保持 logic clock 特点的同时，逼近真实时间

hlc算法

Initially $l.j := 0; c.j := 0$

Send or local event

$l'.j := l.j;$
$l.j := max(l'.j, pt.j);$
If $(l.j = l'.j)$ then $c.j := c.j + 1$
  Else $c.j := 0;$
Timestamp with $l.j, c.j$

Receive event of message m

$l'.j := l.j;$
$l.j := max(l'.j, l.m, pt.j);$
If $(l.j = l'j = l.m)$ then $c.j := max(c.j, c.m) + 1$
  Elseif $(l.j = l'.j)$ then $c.j := c.j + 1$
  Elseif $(l.j = l.m)$ then $c.j := c.m + 1$
  Else $c.j := 0$
Timestamp with $l.j, c.j$


ChinaUnix
ITPUB
THANKS