开源社
kaiyuanshe

China Open Source Conference
COSCon

2016中国开源年会

China Open Source Conference 2016

时间：2016年10月15日-16日

地点：北京航空航天大学

Building a Reliable Large-Scale Distributed Database

Principles and Practice

shenli @ PingCAP

About Me

- Shenli 申砾
- VP of Engineering at PingCAP
- 有道词典 / 360 搜索
- TiDB Tech Leader

Database system nowadays

- Not only human, but also devices (IoT)
  - Big data
- Sharding is painful
- SQL => NoSQL => NewSQL
  - Programming paradigm is changing

What we need?

NewSQL:

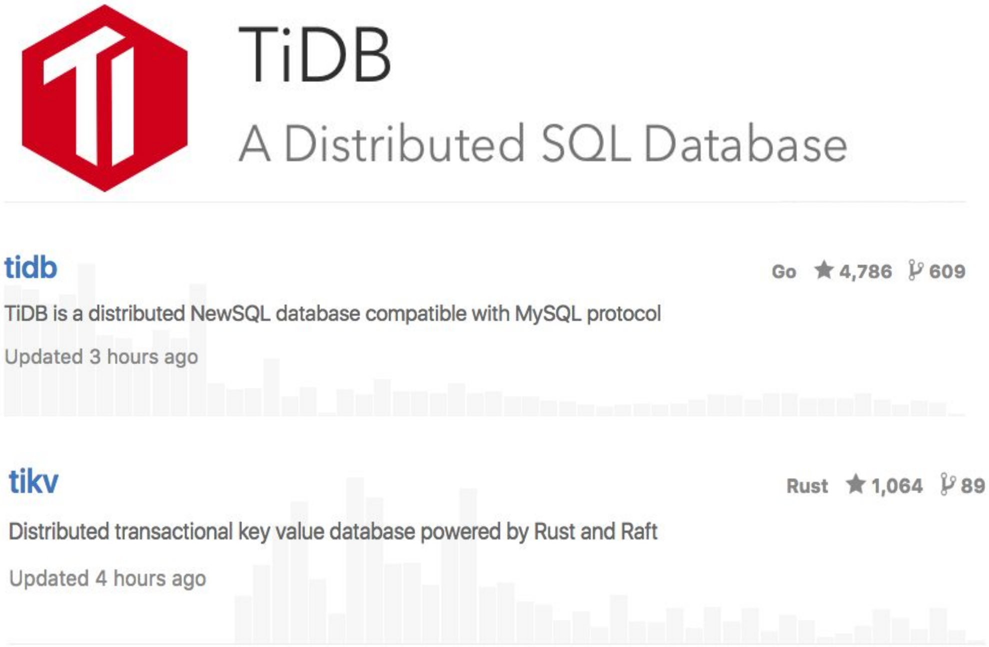
- Scalability is first-class citizen
- SQL
- ACID Transaction
- **Auto-failover / Self recovery / Survivable**

NewSQL is a class of modern relational database management systems that seek to provide the same scalable performance of NoSQL systems for online transaction processing (OLTP) read-write workloads while still maintaining the ACID guarantees of a traditional database system.

-- From Wikipedia

So we build TiDB

- Horizontal scalability
- Consistent distributed transactions
- Compatible with MySQL protocol
- Based on Google Spanner / F1
- One of the most popular open source NewSQL database all over the world



TiDB

A Distributed SQL Database

tidb — Go ★ 4,786 ⑂ 609

TiDB is a distributed NewSQL database compatible with MySQL protocol

Updated 3 hours ago

tikv — Rust ★ 1,064 ⑂ 89

Distributed transactional key value database powered by Rust and Raft

Updated 4 hours ago

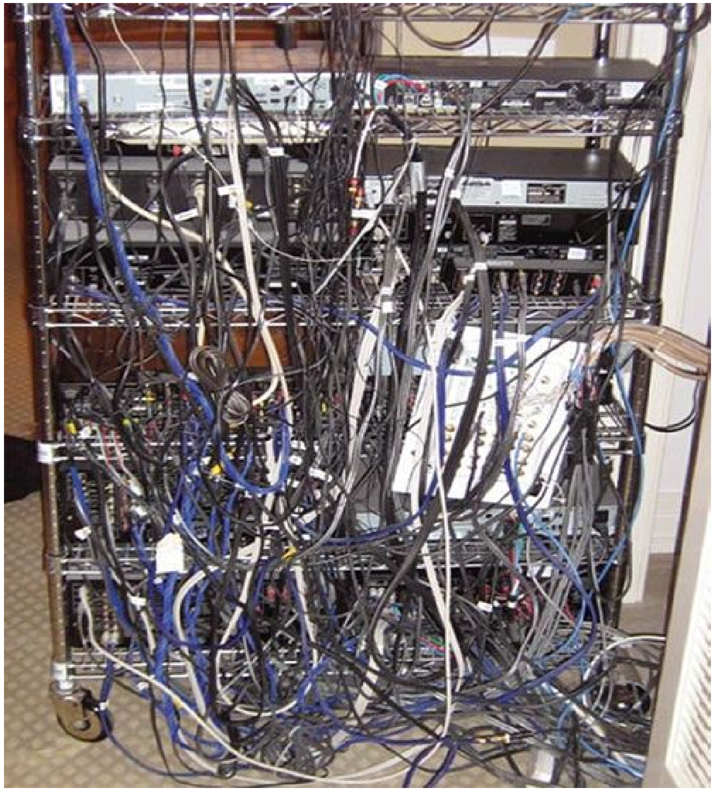
Rule #1:

Always believe shit is about to happen

You might assume...

- Network is reliable and homogeneous
- Latency is small and stable
- Bandwidth is infinity
- Machine is never down
- System administrator is always online
- Process will never be killed
- The whole IDC will never down

But...



WELCOME TO THE REAL WORLD
memegen.com

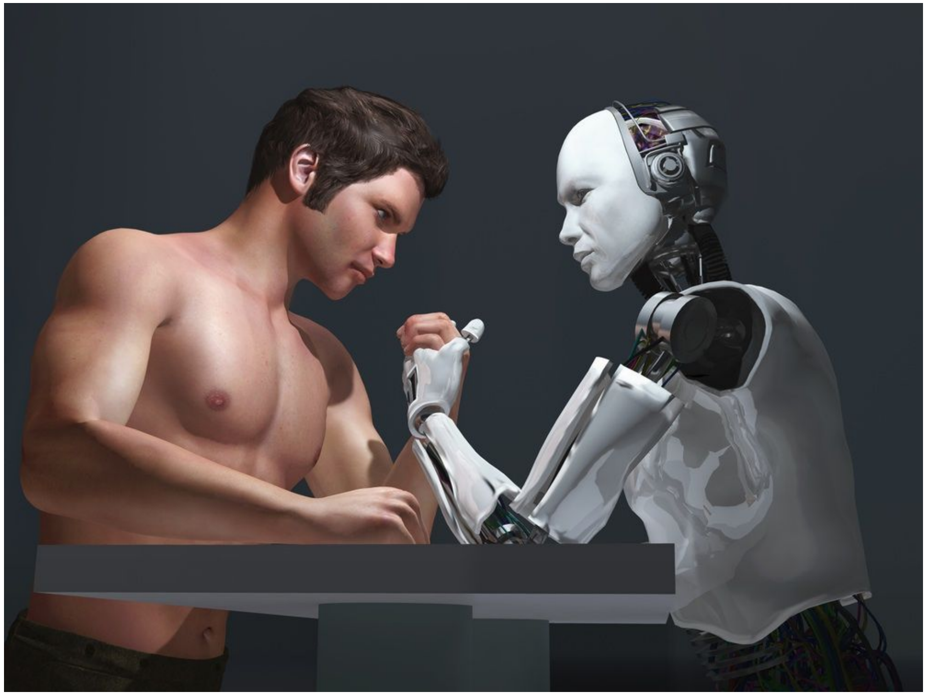
Design for disaster recovery

- Replica matters
- Master-slave is not that reliable!
- Multi-Paxos / Raft saves the world

Rule #2:

Don't rely on humans

Machine won't be tired, people would.



Automate everything

- Auto-scale
- Auto-failover
- Auto-deployment

Auto-scale: Sharding strategy

- Hash-based partitioning
  - e.g. Redis cluster, Codis, Cassandra…
  - Good for balancing workload, but no way to do range scan or prefix seek
- Range-based partitioning
  - e.g. HBase, Spanner, TiDB
  - Good for scanning and prefix seeking, but hard to balance work load

Auto-scale: Balance strategy

- Add replica => Block write => Delete old data
  - Blocking problem
- Non-blocking data transfer
  - e.g. Codis
- Raft
  - Membership change

Master-Slave is not that reliable

- Recovery from network isolation
  - How to detect a network isolation?
  - How to decide which slave should be promoted?
  - You may lose data

Raft / Multi-Paxos in a nutshell

- Replicated state machine
  - Highly autonomous
- Based on election (quorum always wins)
- Just remember:
  - **It never lose any data and it’s automatic.**

Rule #3:

Talk is cheap, show me the tests

Test matters and it's complex

- The hardest part of building a database is how to test it
- It is even harder for a distributed database
- How can we make sure that our code is correct?
- How can we make sure that out pull request is correct?
- Will the new commit slow down the performance?
- What will happen when machine failed?

How to test

- Unit tests
- MySQL tests
- ORM tests
- Home made tests
  - Transfer test, Block write test, Stability test
- Performance tests
- Tests with fault injection
- All tests should be run automatically

Tools

- Jepsen
- Namazu
  - ZooKeeper:

    Found ZOOKEEPER-2212, ZOOKEEPER-2080 (race): (blog article)
  - Etcd:

    Found etcdctl bug #3517 (timing specification), fixed in #3530. The fix also resulted a hint of #3611

    Reproduced flaky tests {#4006, #4039}
  - YARN:

    Found YARN-4301 (fault tolerance)， Reproduced flaky tests{1978, 4168, 4543, 4548, 4556}

Rule #4:

"All problems in computer science can be solved by another level of indirection"

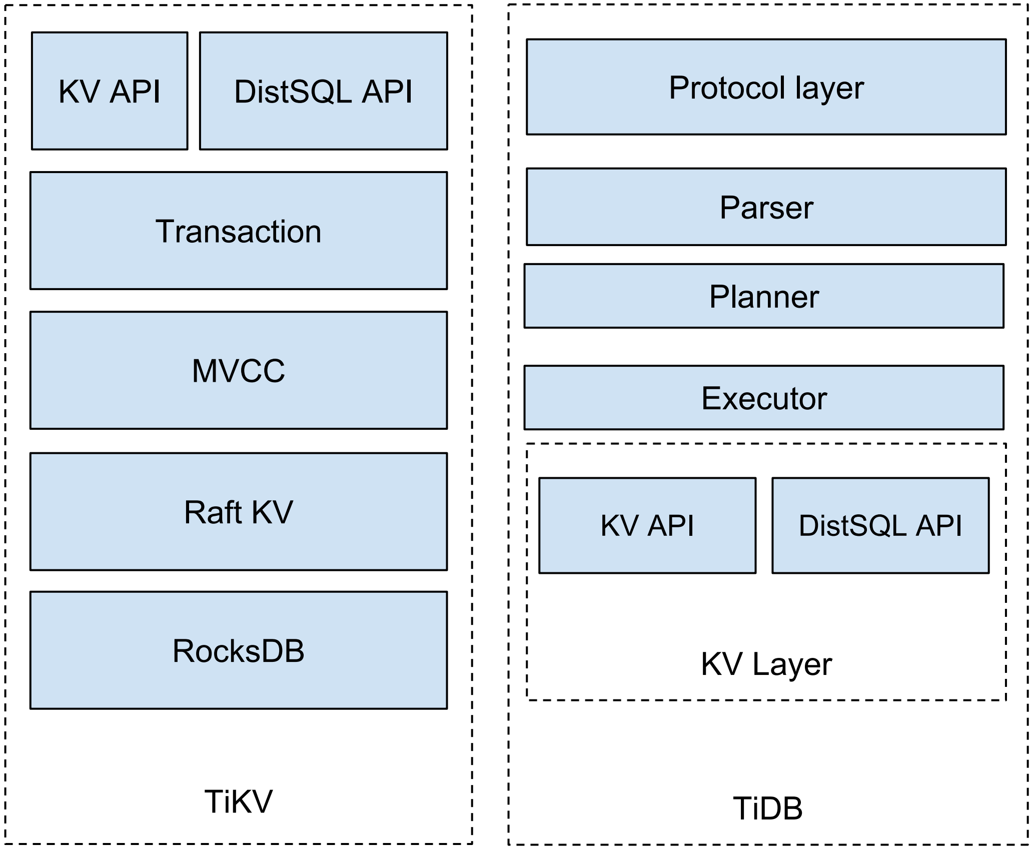
--- David Wheeler

Fighting complexity

- A story about TiDB and TiKV
- SQL layer on a memdb
- SQL layer on local storage
- SQL layer on Hbase as storage engine
- SQL layer on TiKV as storage engine

Highly layered

- Focus on one thing, and do it well.
  - SQL
  - KV
  - Metadata storage
  - Data placement



Rule #5:

Don't try to teach your user, just follow them

Middleware vs NewSQL from scratch

- What’s wrong with middleware?
  - Cross node transaction
  - Global consistency snapshot
  - Cross node join
  - Optimized query plan
  - Seamless horizontal scale
- So let’s start from scratch

Compatibility matters

- User don’t like change.
- User often writes shit code.

Rule #6:

Make it right, and then make it fast

How we make TiDB right and fast

- We build a runnable database in less than a month
- Then we add test framework
- We focus on correctness and elasticity , not performance in the early days
- Strict code review makes sure that we have good architecture and abstraction
- Premature optimization is the root of all evil. --Donald Knuth

Rule #7:

Embrace the community you don't need to do everything

Open source matters

- We want to be part of the whole big data environment.
- No vendor lock-in.
- Professional guys handle professional things.
- Use standard interfaces such as SQL, Binlog.
- Contribute to the open source community and enjoy the benefit from the open source community
    - tidb/parser
    - rust-prometheus/rust-grpc
    - go-mysql, go-hbase
    - etcd, rocksdb

Open source tools

- Storage Engine
  - facebook/rocksdb
  - liblz4
- Monitoring
  - Prometheus
  - Grafana
- Deployment
  - Docker
  - Kubernetes

Q&A

TiDB: http://github.com/pingcap/tidb （4700+ stars）
TiKV: https://github.com/pingcap/tikv （1000+ stars）

My Wechat

PingCAP Official Accounts