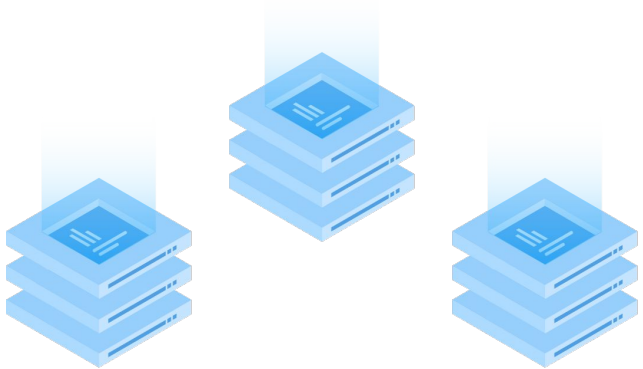
TiKV: A Cloud-Native Key-Value Database

Presented by Dongxu Huang and Nick Cameron

KubeCon + CloudNativeCon EU August 2020

TiKV

CLOUD NATIVE
COMPUTING FOUNDATION



About us

- Dongxu (Ed) Huang
- Co-founder & CTO, PingCAP, TiDB/TiKV
- Database geek, distributed system engineer
- Rust / Golang bilingual
- Twitter: @dxhuang
- Email: huang@pingcap.com

About us

- Nick Cameron
- Senior engineer at PingCAP
- TiKV contributor
- Rust core team
- Twitter: @nick_r_cameron
- GitHub: @nrc

PingCAP

Introduction and History

A little bit of history

PingCAP, TiDB and TiKV

In the spring of 2015, after reading papers on Google Spanner and F1, 3 engineers decided to launch their own startup to build an open source implementation of Spanner (of MySQL protocol)

Spanner: Google's Globally-Distributed Database

James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, JJ Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, Dale Woodford

Google, Inc.

Abstract

Spanner is Google's scalable, multi-version, globally-distributed, and synchronously-replicated database. It is the first system to distribute data at global scale and support externally-consistent distributed transactions. This paper describes how Spanner is structured, its feature set, the rationale underlying various design decisions, and a novel time API that exposes clock uncertainty. This API and its implementation are critical to supporting external consistency and a variety of powerful features: non-blocking reads in the past, lock-free read-only transactions, and atomic schema changes, across all of Spanner.

F1: A Distributed SQL Database That Scales

Jeff Shute, Chad Whipkey, David Menestrina, Radek Vingralek, Eric Rollins, Stephan Ellner, Traian Stancescu, Bart Samwel, Mircea Oancea, John Cieslewicz, Himani Apte, Ben Handy, Kyle Littlefield, Ian Rae*

Google, Inc.
*University of Wisconsin-Madison

ABSTRACT

F1 is a distributed relational database system built at Google to support the AdWords business. F1 is a hybrid database that combines high availability, the scalability of NoSQL systems like Bigtable, and the consistency and usability of traditional SQL databases. F1 is built on Spanner, which provides synchronous cross-datacenter replication and strong consistency. Synchronous replication implies higher commit latency, but we mitigate that latency by using a hierarchical schema model with structured data types and through smart application design. F1 also in-

consistent and correct data.
Designing applications to cope with concurrency anomalies in their data is very error-prone, time-consuming, and ultimately not worth the performance gains.

4. **Usability**: The system must provide full SQL query support and other functionality users expect from a SQL database. Features like indexes and ad hoc query are not just nice to have, but absolute requirements for our business.

PingCAP

A little bit of history: SQL First!

This is a very ambitious project for a young startup, and we chose to start with the SQL layer (which is the F1 part).

Why?

- the SQL layer is closer to application and better expresses our ideas in the early days of the project
- More importantly, for quality purposes, we used a lot of test cases from MySQL and its surrounding ecosystem.

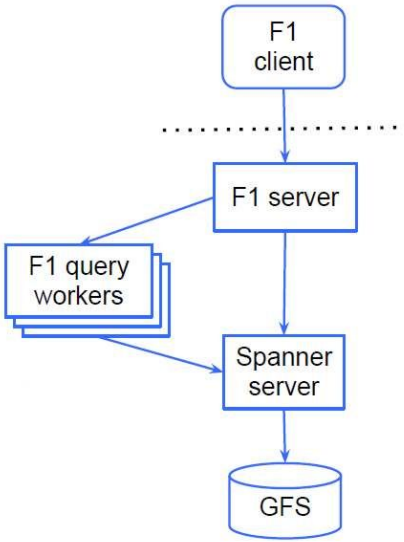
A little bit of history

But when we finished the first version of the TiDB SQL layer, problem arose

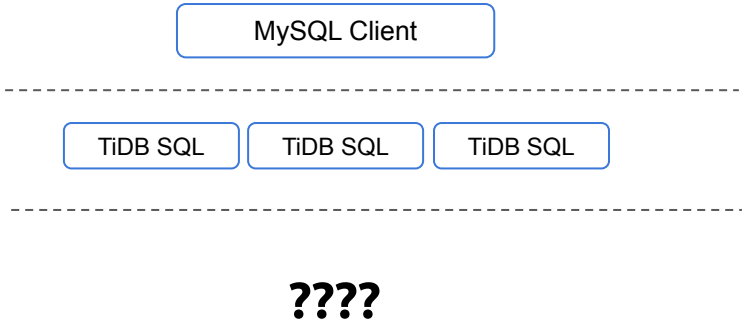
A little bit of history

Who is going to use a database that can't store data?

A little bit of history



This is Google F1



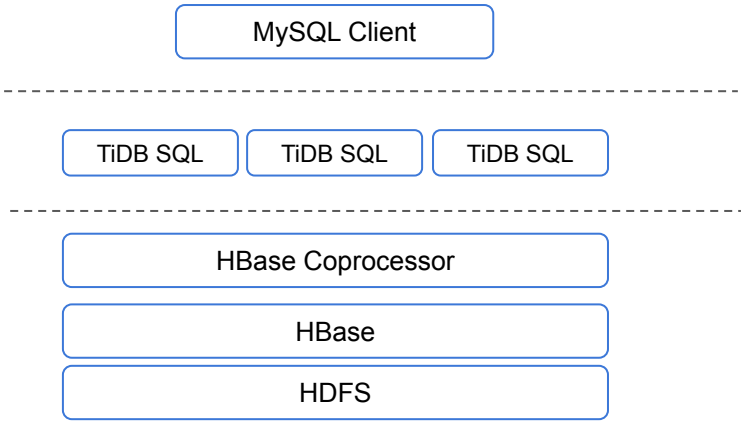
????

This is us (in Sep, 2015)

PingCAP

A little bit of history

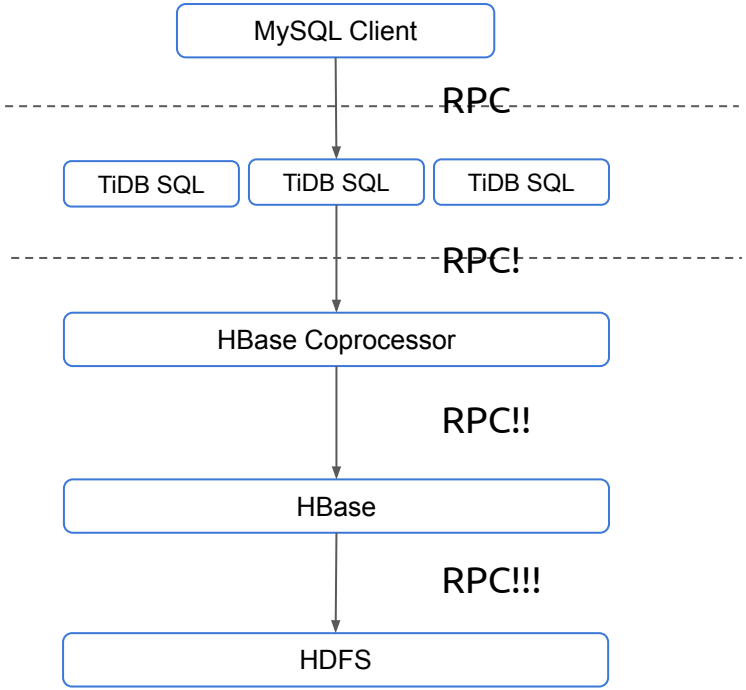
We chose HBase as our first storage engine



This is us (in early 2016)

Looks like a distributed SQL database, right?

A little bit of history



Poor performance & too many dependencies

A little bit of history

- By the end of 2016, We decide to build our own distributed KV layer
- Requirements:
  - Scale-out like other NoSQL systems
  - Performance matters
  - Built-in ACID transaction support
  - Modern data replication protocol
  - Fewer layers of abstraction
- Target:
  - Building block (storage) for other distributed systems

A little bit of history

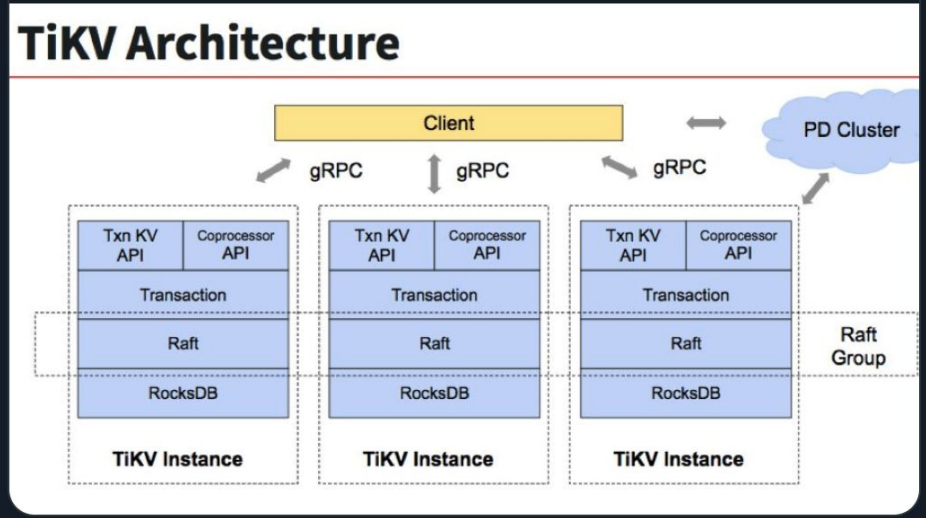
- By the end of 2016, We decide to build our own distributed KV layer
- Requirements:
  - Scale-out like other NoSQL systems (Auto sharding/scaling)
  - Performance matters (Rust + RocksDB)
  - Built-in ACID transaction support (2PC, but also supports atomic KV API)
  - Modern data replication protocol (Raft & Multi-Raft)
  - Fewer layers of abstraction (No need for DFS, thanks to Raft)
- Target:
  - Building block (storage) for other distributed systems (Transactional KV store)

TiKV

Ed Huang @dxhuang · Aug 29, 2018

Before creating **TiKV**, I have always dreamed of having a high-performance key-value database that supports ACID transaction, then I can build many systems without pain, I hope to see community considers **TiKV** as the **building block** to other interesting distributed systems, enjoy it!

> **Chris Aniszczyk** @cra · Aug 28, 2018
>
> ⚡ happy to welcome the TiKV project to the @CloudNativeFdn sandbox! this marks our 2nd project born in China and our 2nd project that is primarily written in @rustlang cncf.io/blog/2018/08/2...
>
> Show this thread



2 12 38 Tip

PingCAP

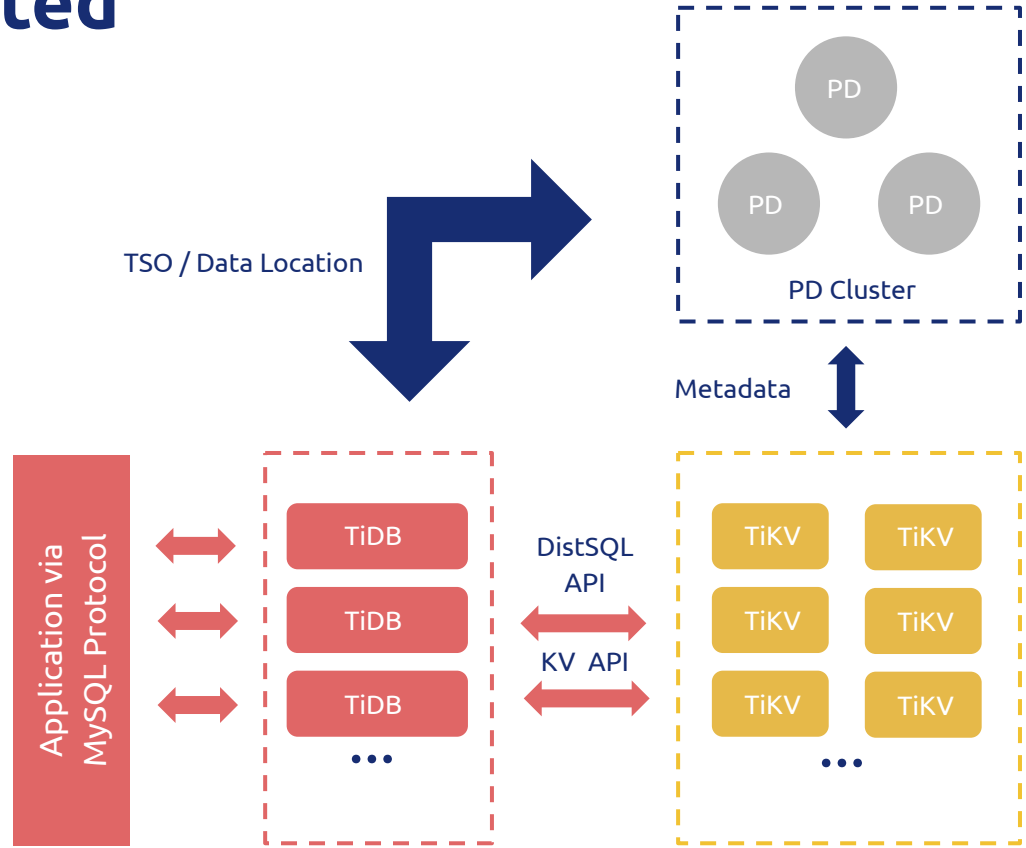
PingCAP

Architecture

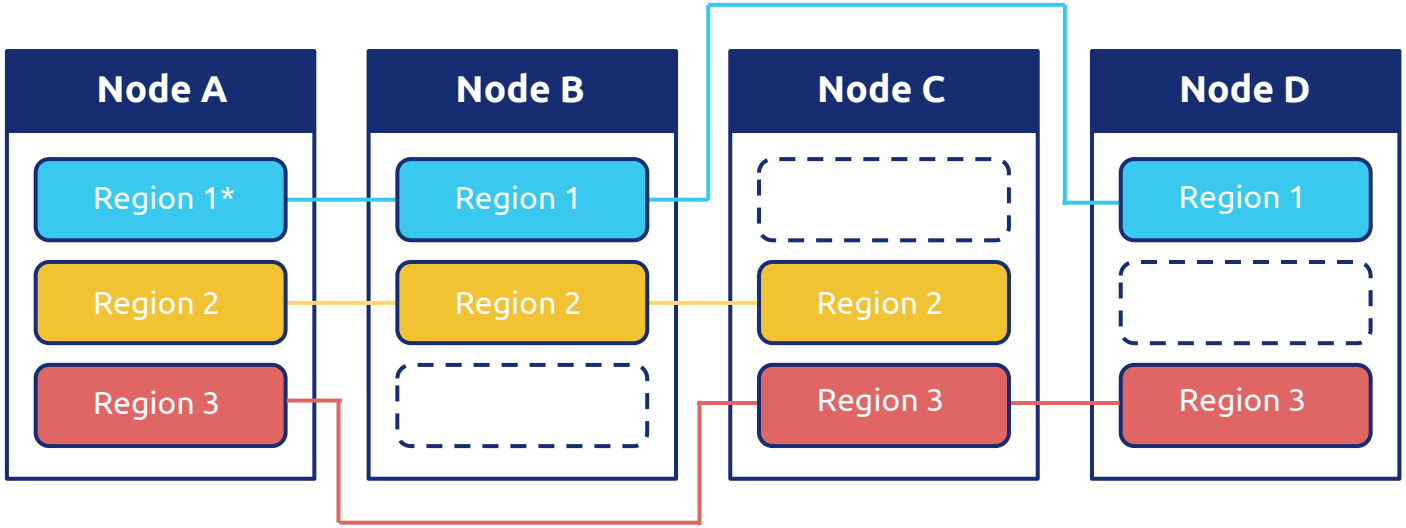
TiKV is

- Distributed
- Transactional
- Key-value store

Distributed



Distributed



TiKV is

Transactional API

MVCC

Raft

Rocks DB

Coprocessor

PingCAP

Transactional

- ACID
- Snapshot isolation, repeatable reads, linearizable

Transactional

- Collaborative
- MVCC

Reliability

- Regression and performance testing
- Jepsen
- Chaos Mesh
- TLA+

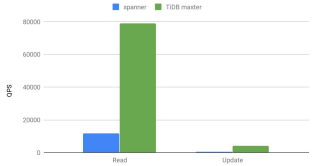
Benchmarks

- YCSB
- TiDB c.f. Spanner
- Google cloud, $2100-2300, end of 2019

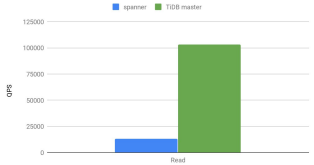
Benchmarks

workloada-QPS

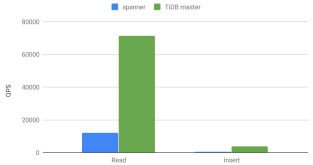
workloadb-QPS



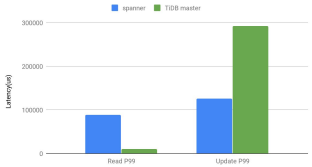
workloadc-QPS



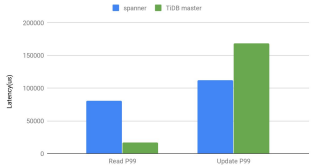
workloadd-QPS



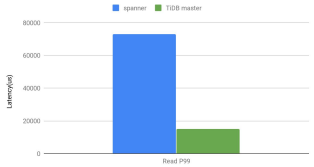
workloada-latency



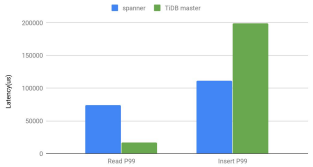
workloadb-latency

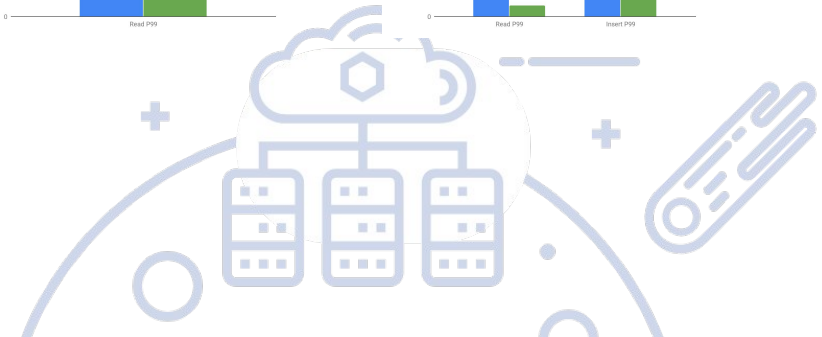


workloadc-latency



workloadd-latency





PingCAP

PingCAP

Looking forward

Flexibility

- Plugable storage engine
- Versioned API

Stability

- Latency-based flow control
- Auto-scaling
- Improve latency jitter

Performance

- SATA SSD optimizations
- Memory usage
- Raft joint consensus

PingCAP

Community

Community

- Open source since 2016
- CNCF incubating project

Community

- SIGs
  - Engine
  - Raft
  - Transactions
  - Coprocessor

Community

https://github.com/tikv/tikv

tikv-wg

@nrc

https://tikv.org/chat

PingCAP

Thank You !