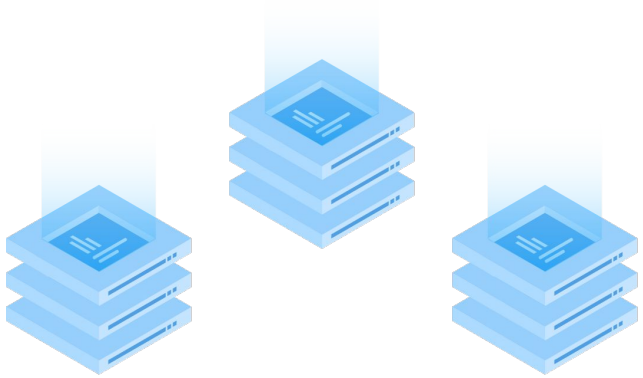
How Does Geo-replication Work in TiDB

Presented by Jay Lee

PingCAP TiDB



About Me

Jay Lee (李建俊)
Senior Engineer of PingCAP

Distributed system engineer / Database engineer/
Open- source advocator
TiKV Maintainer
Rustacean: raft-rs / grpc-rs

Github @busyjay

PingCAP TiDB

Overview

- What is TiDB
- How does replication work
- Deployment
- Q&A

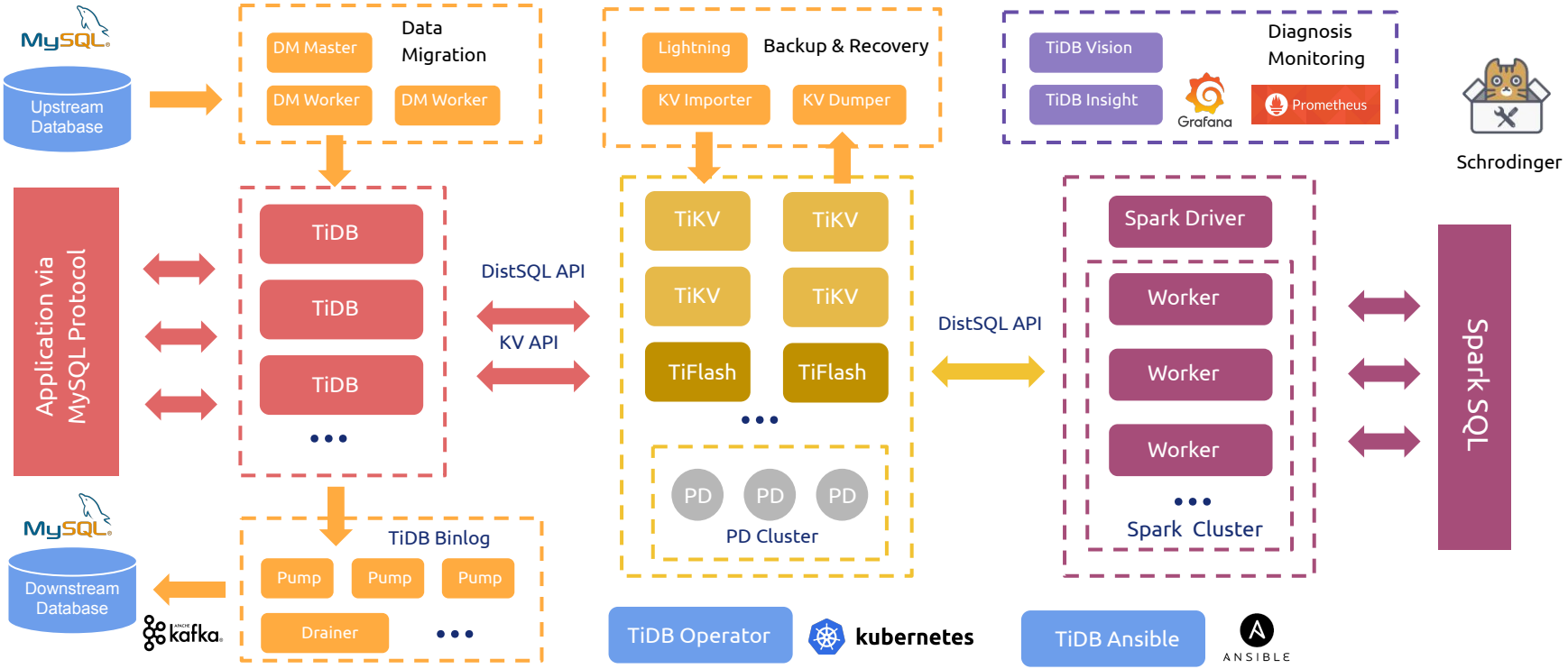
PingCAP

Part I - Intro to TiDB

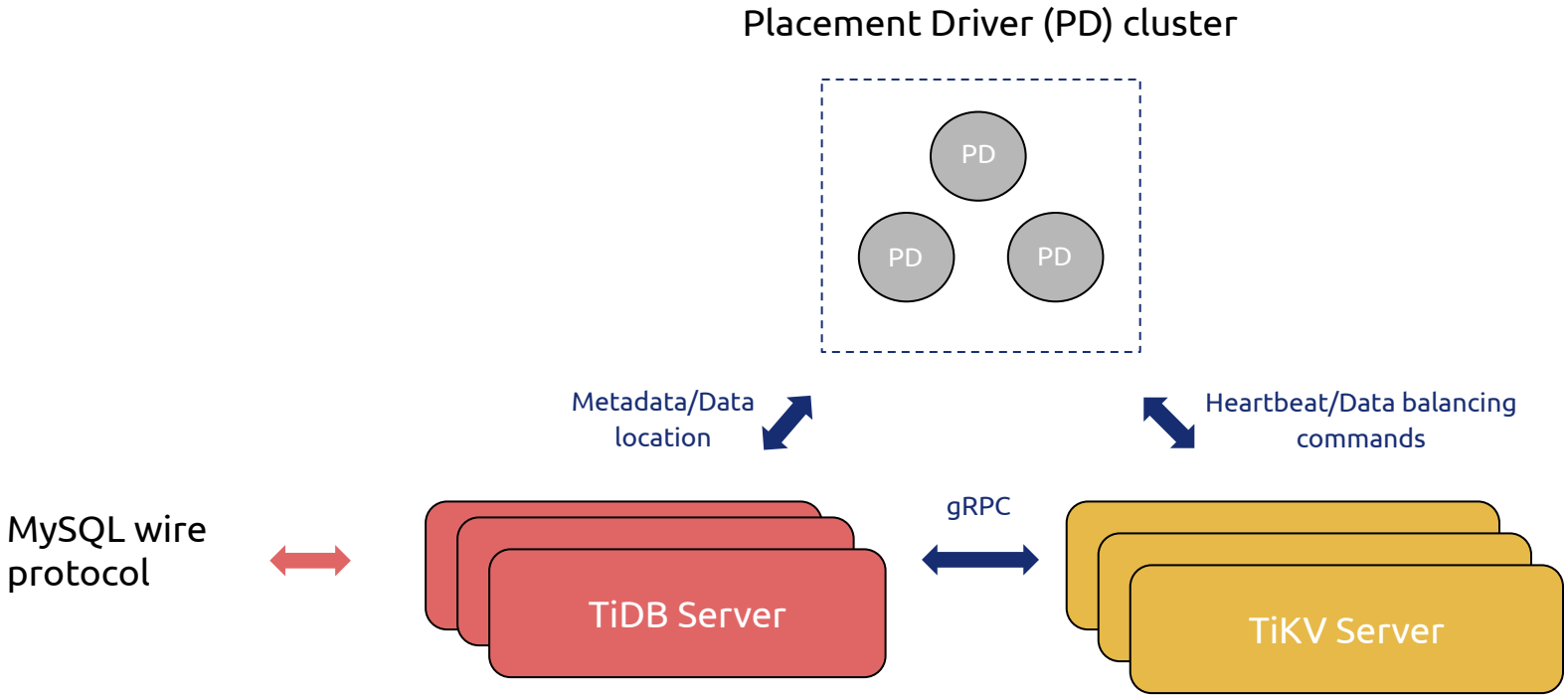
What is TiDB

- Elastic scaling-out
  - Transparent to applications, no more manual sharding!
- Always-on
  - HA with strong consistency
- SQL
  - MySQL dialect
  - HTAP = OLTP + OLAP
- ACID semantics

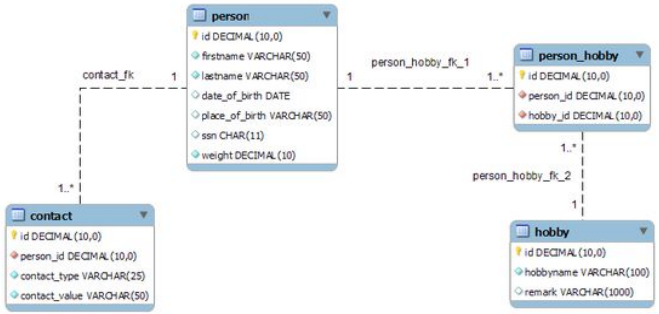
The whole picture



A closer review



Data organization within TiDB



Tables (Rows with columns)

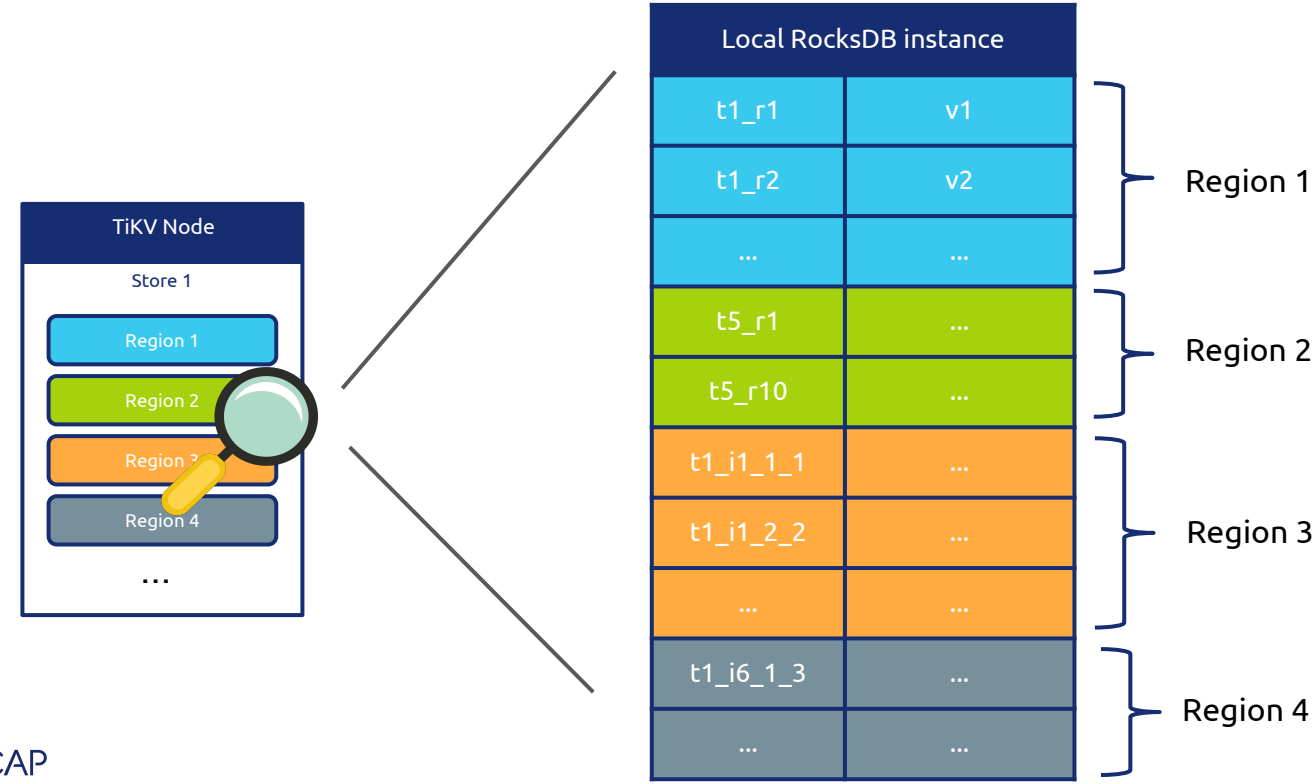
tidb-server

Dictionary order

TiKV	
t1_r1	{col1: v1, col2: v2, col3: v3 ...}
t1_r1	{col1: v1, col2: v2, col3: v3 ...}
...	...
t2_r1	{col1: v1, col2: v2 ...}
...	...

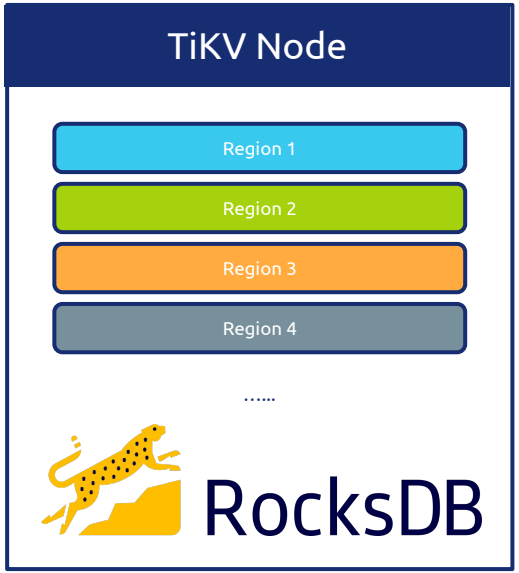
Key-value pairs within TiKV ordered by key

Data organization within TiDB

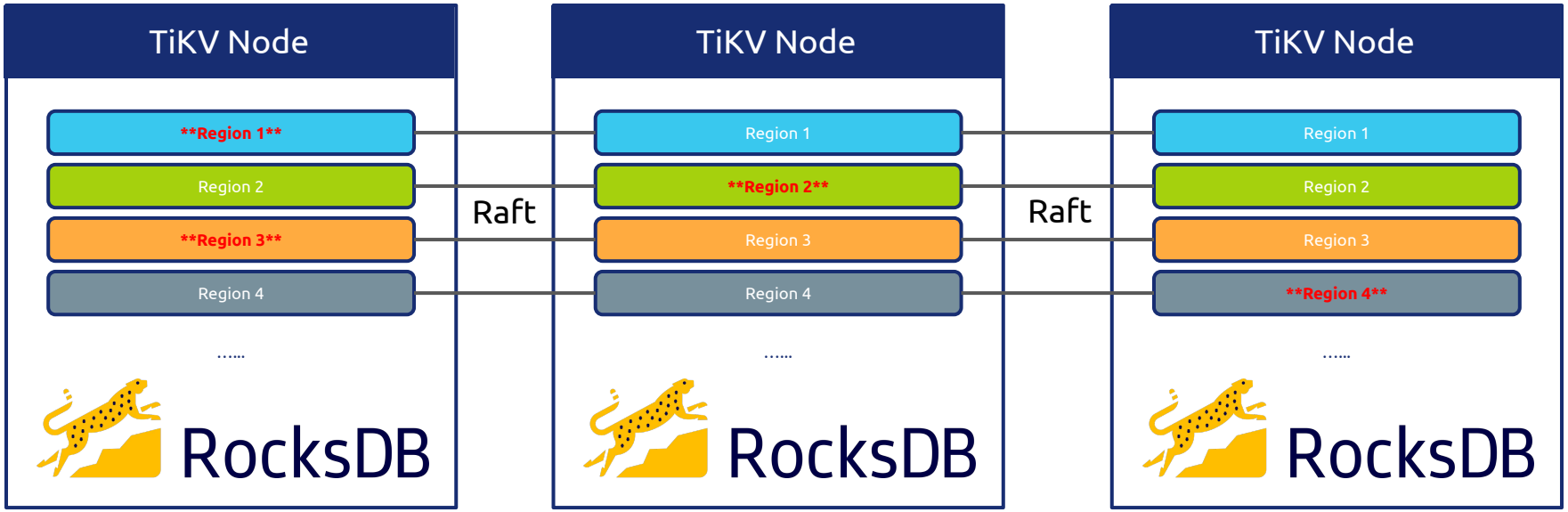


Data organization within TiDB

- Region (A bunch of key-value pairs, or Split)
  - Default total size of one Region: 96MB
    - You can change it in configuration
- Region is a logical concept
  - Region meta : [Start key, End key)
  - All regions within a TiKV node share a same **RocksDB** instance
- Each Region is a Raft group
  - Default: 3 replicas



Region: multiple replicas across different nodes

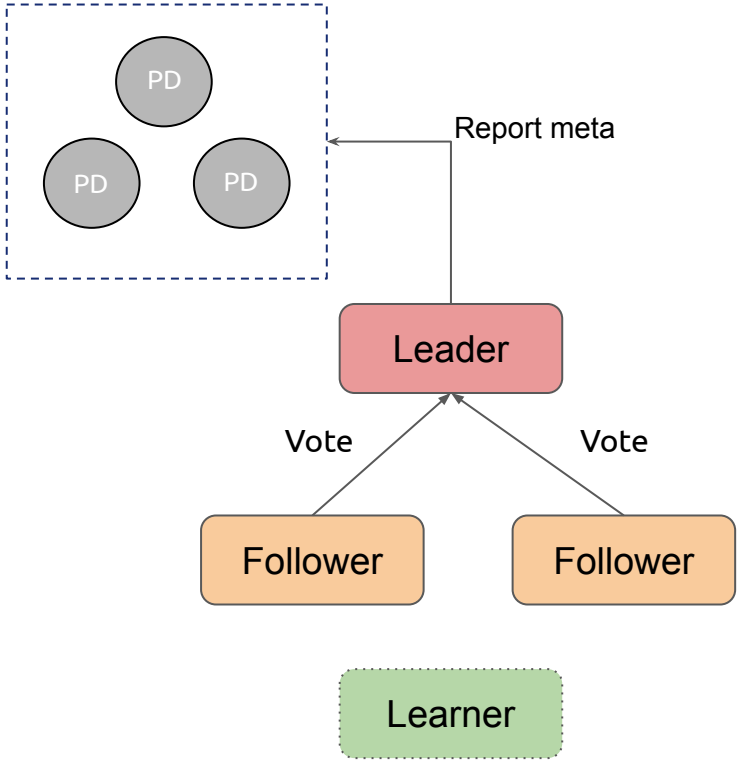


PingCAP

Part II - How does replication work

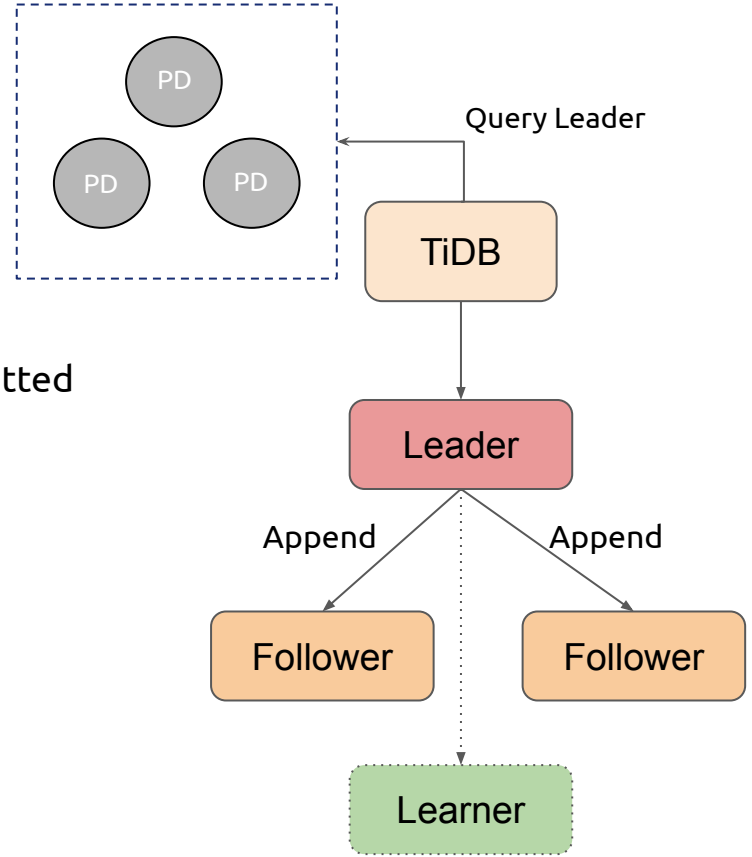
Raft group

- A region is managed by a raft group
- Possible roles
  - Leader
  - Follower
  - Learner
- Leader receives votes from majority followers
- Leader manages group and report to PD
- On failure
  - Follower starts campaign and take over



Write

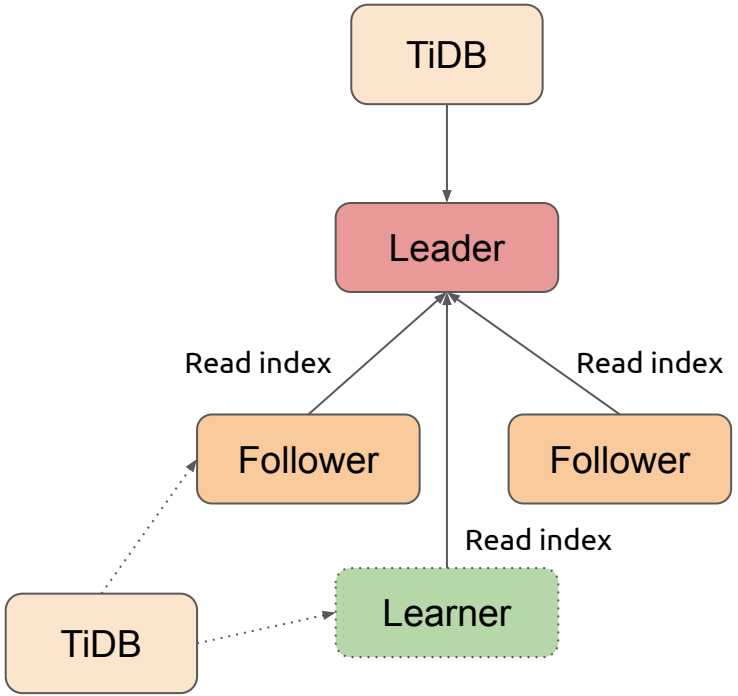
- Data is written to leader as logs
- Leader replicates logs to followers and learners
- Logs replicated to majority followers are committed



PingCAP TiDB

Read

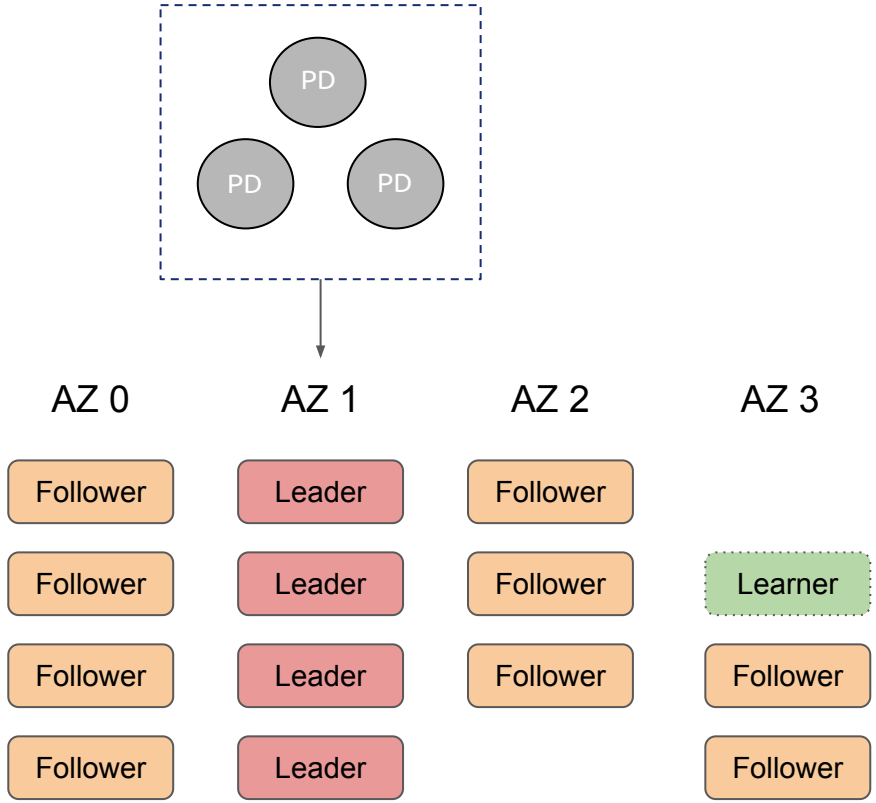
- All roles can read
- Read on leader
  - Read immediately if in lease
  - Renew lease otherwise
- Read on follower and learner
  - Read log index on leader
  - Read data on follower



PingCAP TiDB

Configuration change

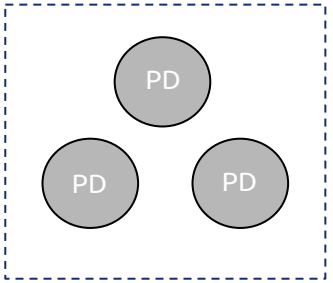
- Simple change
  - Can only handle one change at a time
  - Process as a special write
  - Change in one step
  - Quick and easy

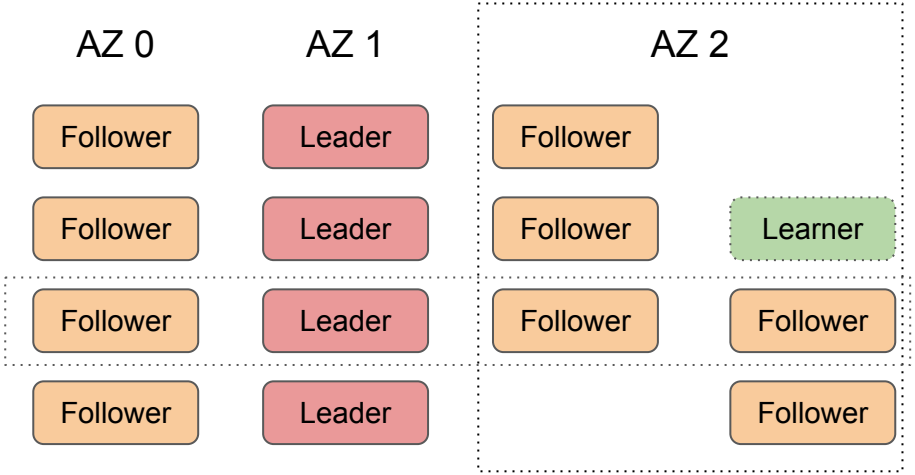


PingCAP TiDB

Configuration change

- Simple changes can lead to unavailability

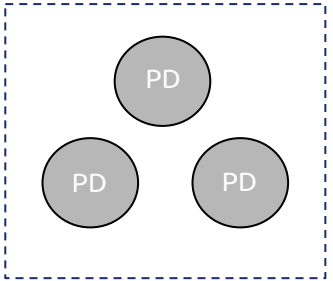


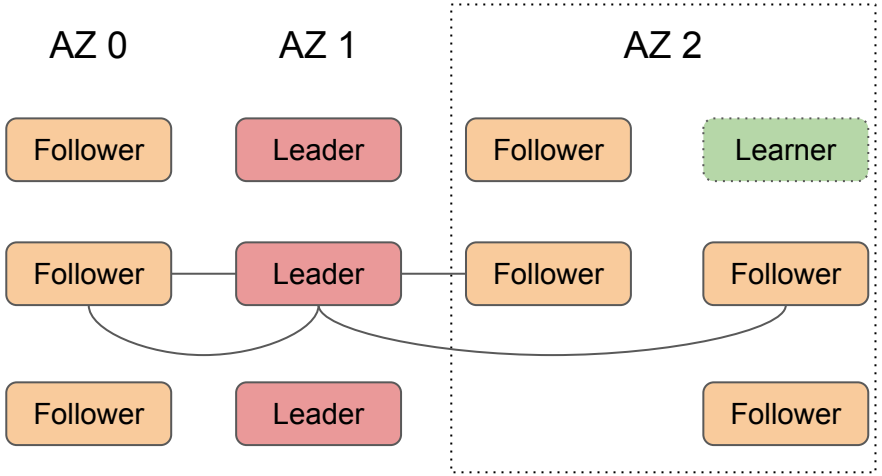


PingCAP TiDB

Configuration change

- Simple changes can lead to unavailability
- Joint Consensus
  - Enter joint state first
    - Both new and old configuration takes effects
  - Complicated

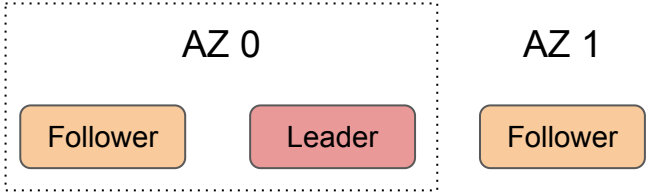




PingCAP TiDB

Commit group

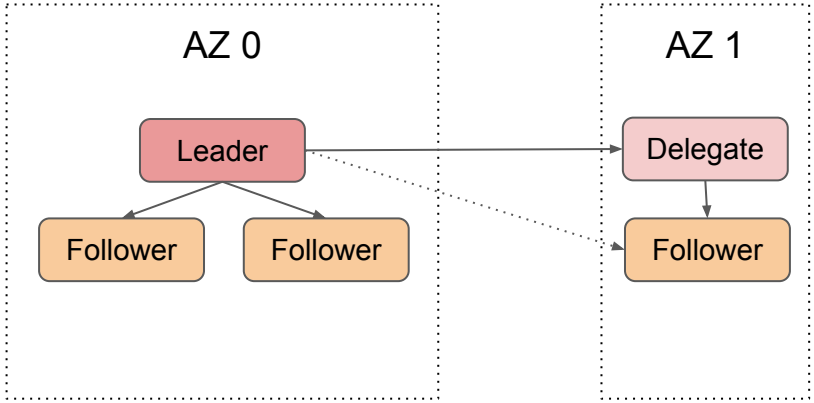
- Majority may not ensure data safety
  - Destroy of one AZ can lost writes
- Replicas are assigned to different groups
- Group is calculated according to AZs
- Logs commit
  - Majority from configuration
  - Replicated to at least 2 groups



PingCAP TiDB

Commit group

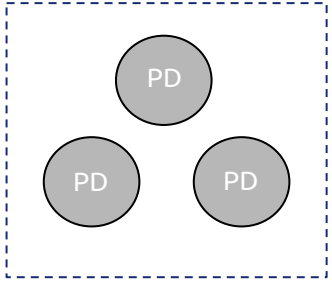
- A delegate is assigned to a group
  - Forwarding logs from leader
- Reduce bandwidth by half

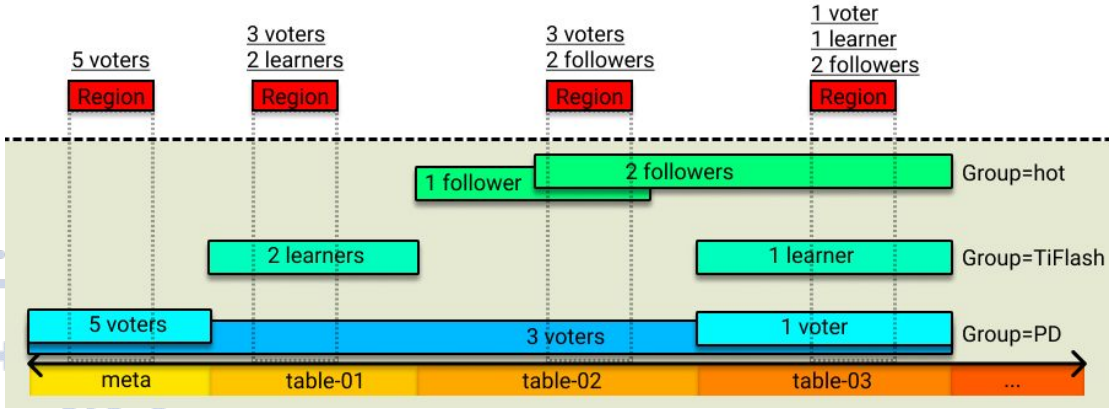


PingCAP TiDB

Placement rules

- Place replicas by rules
- Rule
  - Contstraints on replication numbers
  - Raft roles
  - Geolocation
- Work on ranges





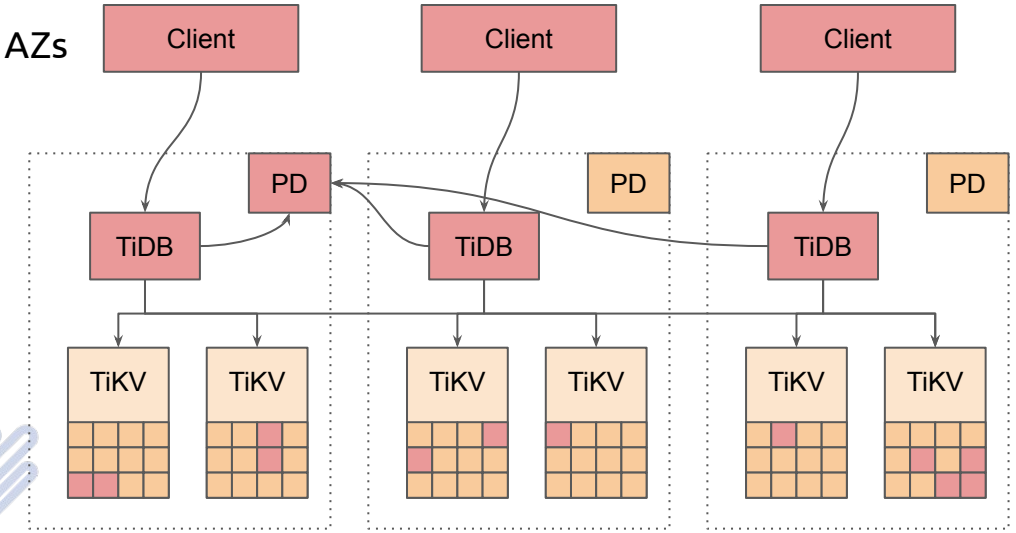
PingCAP TiDB

PingCAP

Part III - Deployment

Three AZs

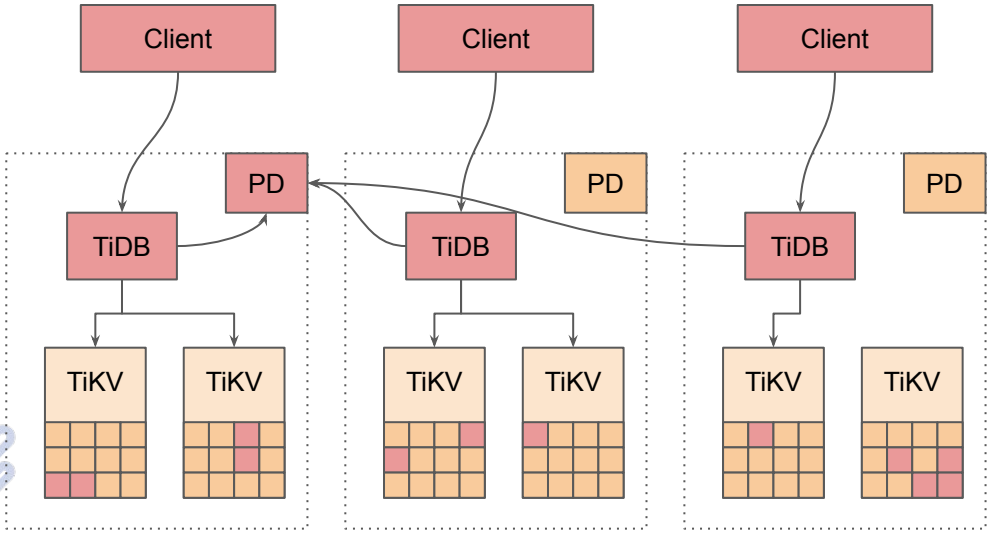
- Replicas and leaders are distributed among 3 AZs
- Tolerate one down AZ
- Disavantage
  - High latency when request across AZs



PingCAP TiDB

Three AZs

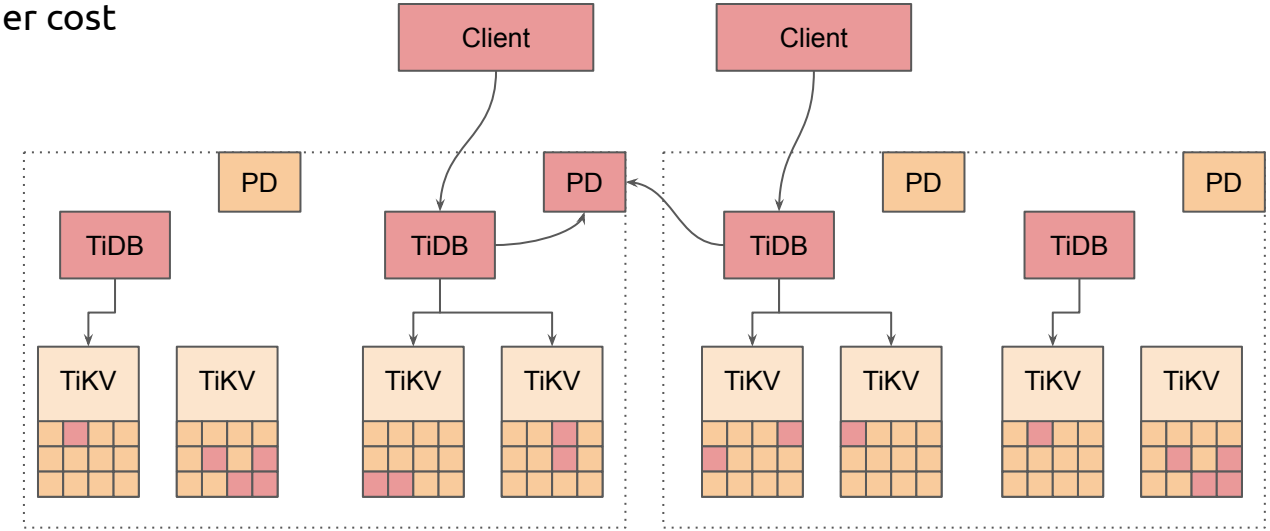
- Replicas are distributed among 3 AZs
- Placement rules in PD
  - controls leaders and replicas
- Leaders are scheduled relative to client
- Read can be optimized by follower read



PingCAP TiDB

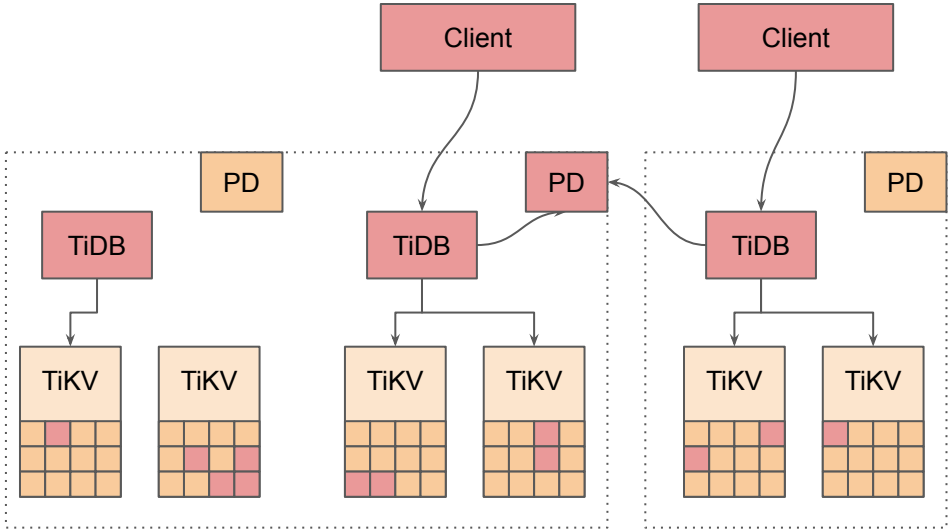
Two AZs

- Use even number of replicas
- Safety
  - No data lost when either AZ fails
- Availability
  - Unavailable when either AZ fails
- Higher cost



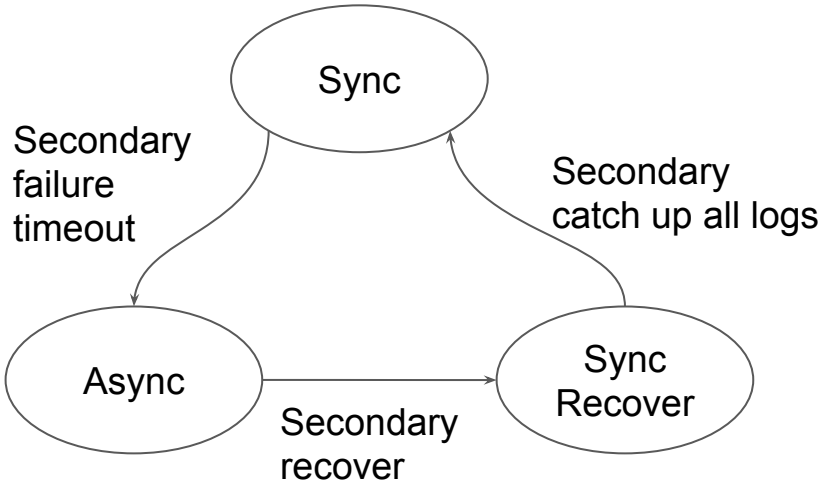
Two AZs

- Odd number of replicas + group commit
- Safety is guaranteed by group commit
- Primary AZ has more replicas
- Failure of secondary AZ can be recover by removing group commit



Two AZs

- PD manages replication states
- Only Sync state guarantees safety



Summary

- Use raft algorithm to ensure atomicity and consistency
- Schedule Leaders to reduce latency using placement rules
- Introduce group commit to ensure safety across even AZs
- Follower read and replication to reduce bandwidth

PingCAP

Thank You !

Twitter: @pingcap @busyjaylee
https://www.pingcap.com
Slack: #everyone in Slack