Deep Dive into TiDB

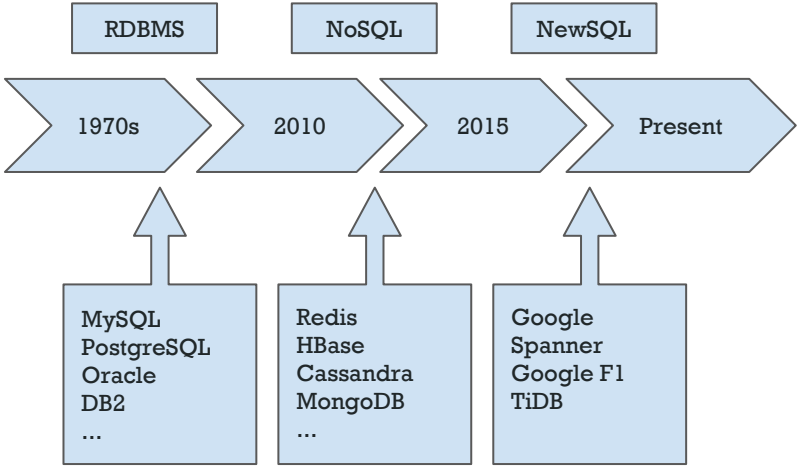
SunHao | PingCAP

Agenda

- Why we need a new database
- The goal of TiDB
- Design && Architecture
  - Storage Layer
  - Scheduler
  - SQL Layer
  - Spark integration
  - TiDB on Kubernetes

Why we Need a NewSQL Database

- From scratch
- What's wrong with the existing DBs?
  - RDBMS
  - NoSQL & Middleware
- NewSQL: F1 & Spanner



What to build?

- Scalability
- High Availability
- ACID Transaction
- SQL

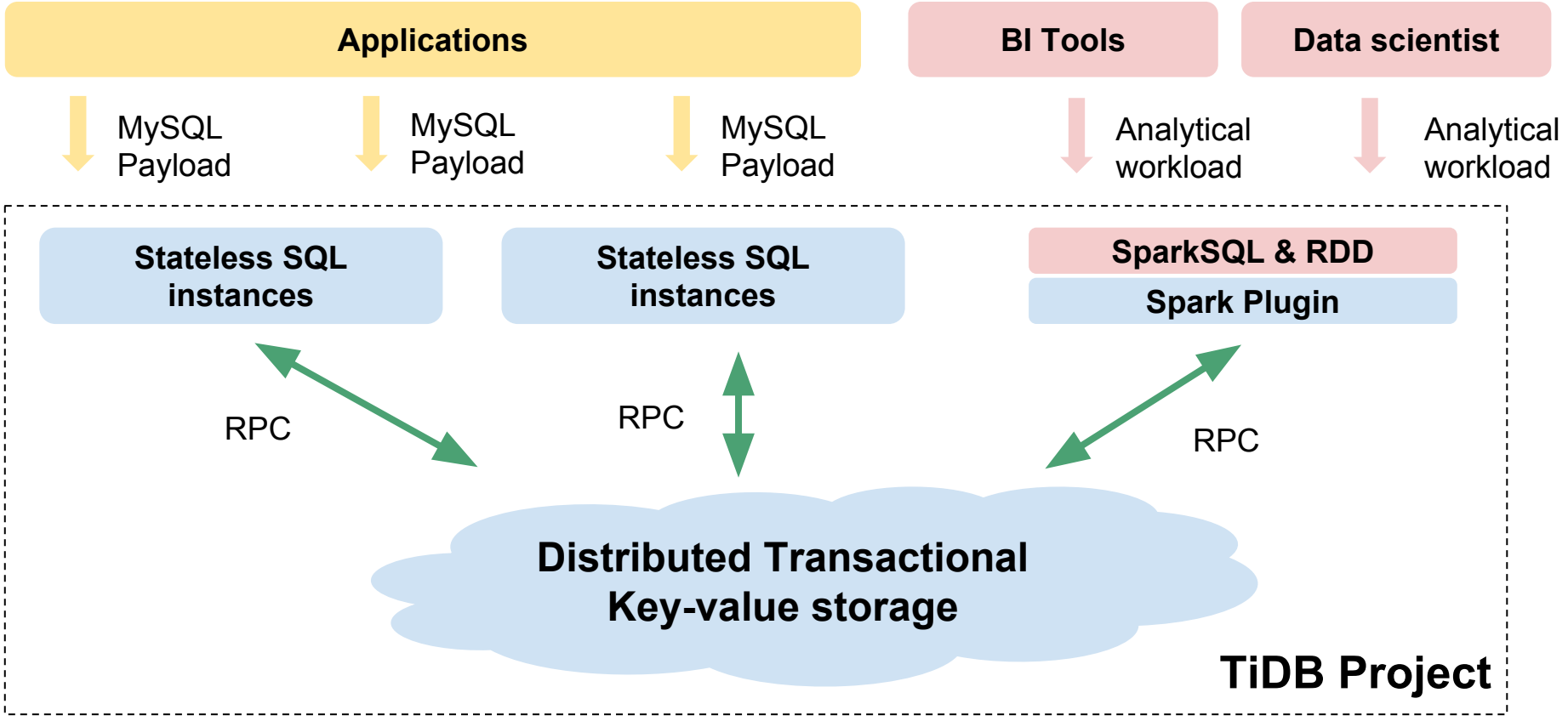
A Distributed, Consistent, Scalable, SQL Database that supports the best features of both traditional RDBMS and NoSQL

Open source, of course

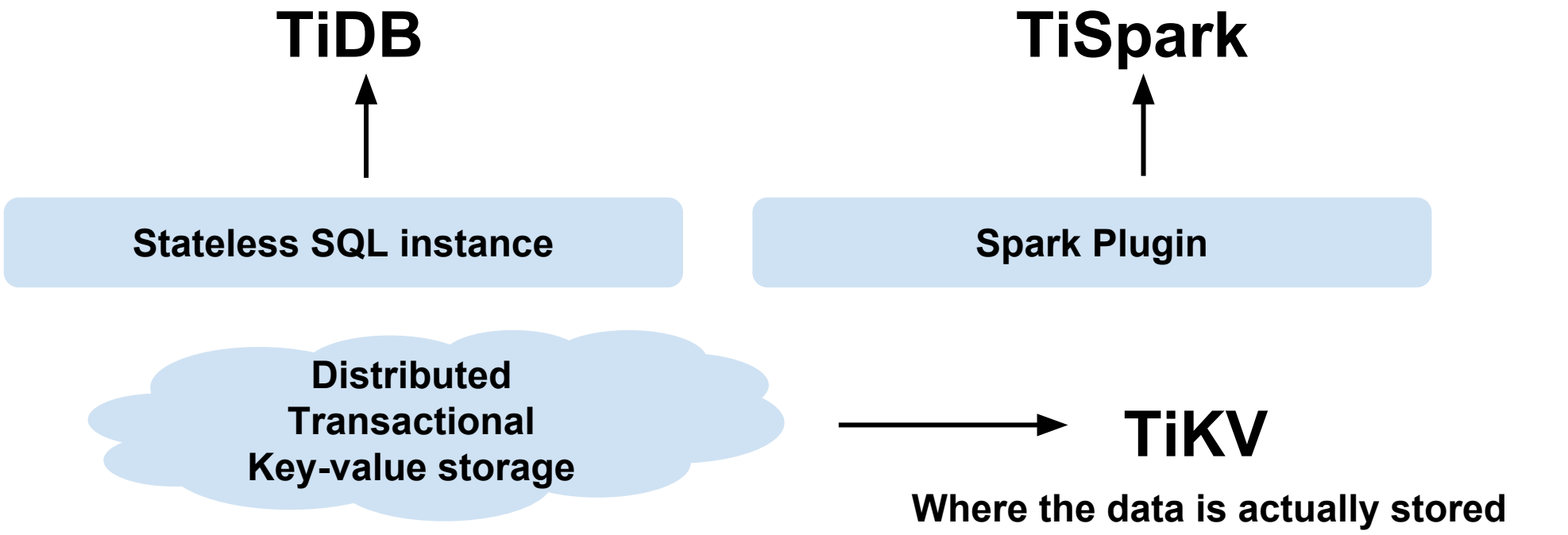
What problems we need to solve

- Data storage
- Data distribution
- Data replication
- Auto balance
- ACID Transaction
- SQL at scale

Overview


Applications
BI Tools
Data scientist
MySQL Payload
MySQL Payload
MySQL Payload
Analytical workload
Analytical workload
Stateless SQL instances
Stateless SQL instances
SparkSQL & RDD
Spark Plugin
RPC
RPC
RPC
Distributed Transactional Key-value storage
TiDB Project

Overview


TiDB
TiSpark
Stateless SQL instance
Spark Plugin
Distributed
Transactional
Key-value storage
TiKV
Where the data is actually stored

TiKV as a KV engine

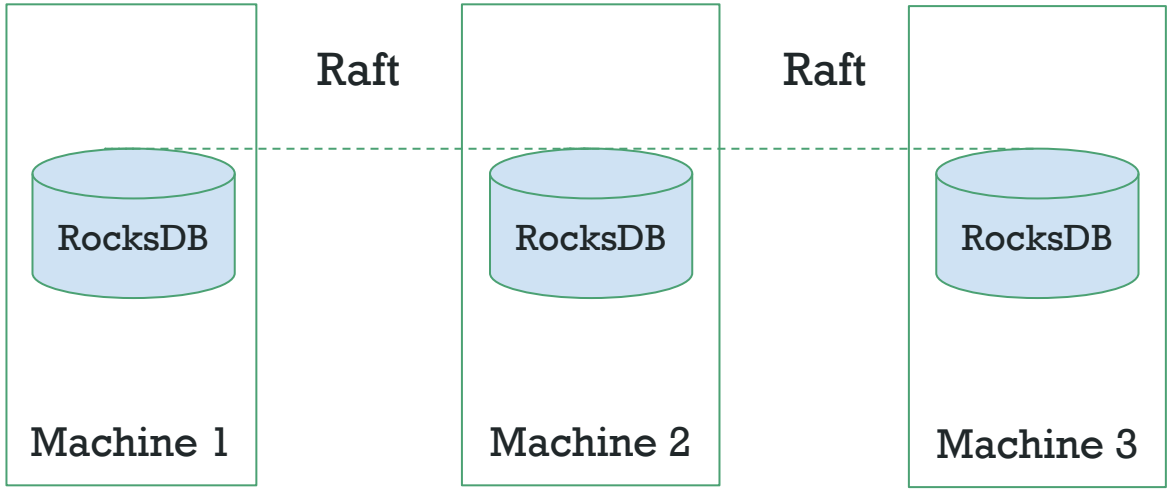
A fast KV engine: RocksDB

- Good start! RocksDB is fast and stable.
  - Atomic batch write
  - Snapshot
- However… It's a locally embedded KV store.
  - Can't tolerate machine failures
  - Scalability depends on the capacity of the disk

Let's fix Fault Tolerance

- Use Raft to replicate data
  - Key features of Raft
    - Strong leader: leader does most of the work, issue all log updates
    - Leader election
    - Membership changes
- Implementation:
  - Ported from etcd
- Replicas are distributed across machines/racks/data-centers

Let's fix Fault Tolerance



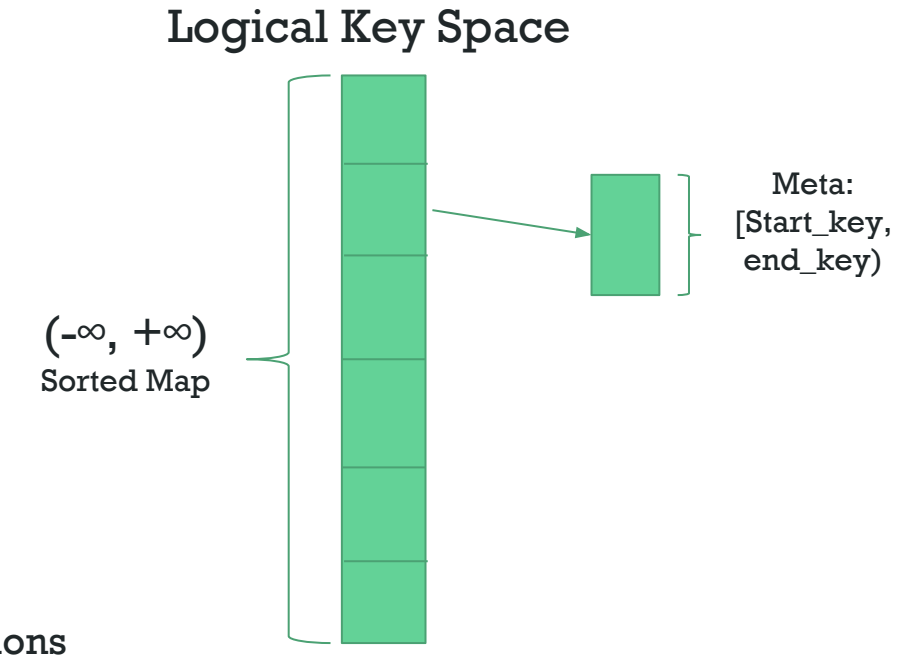
How about Scalability?

- What if we **SPLIT** data into many regions?
  - We got many Raft groups.
  - Region = Contiguous Keys
- Hash partitioning or Range partitioning?
  - Redis: Hash partitioning
  - HBase: Range partitioning

Range Scan:
Select * from t where c > 10 and c < 100;

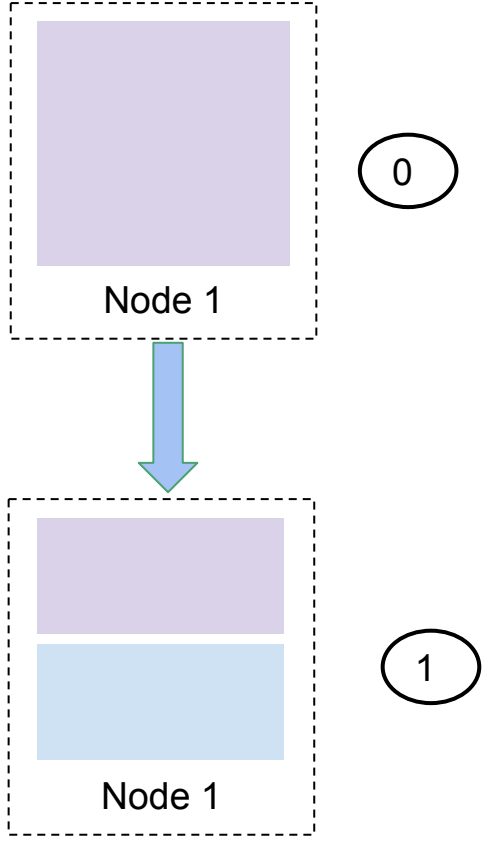
Region

- Key: Byte Array
- A **globally ordered map**
  - Can't use hash partitioning
  - Use **range** partitioning
    - Region 1 -> [a - d]
    - Region 2 -> [e - h]
    - …
    - Region n -> [w - z]
  - Data is stored/replicated/scheduled in regions

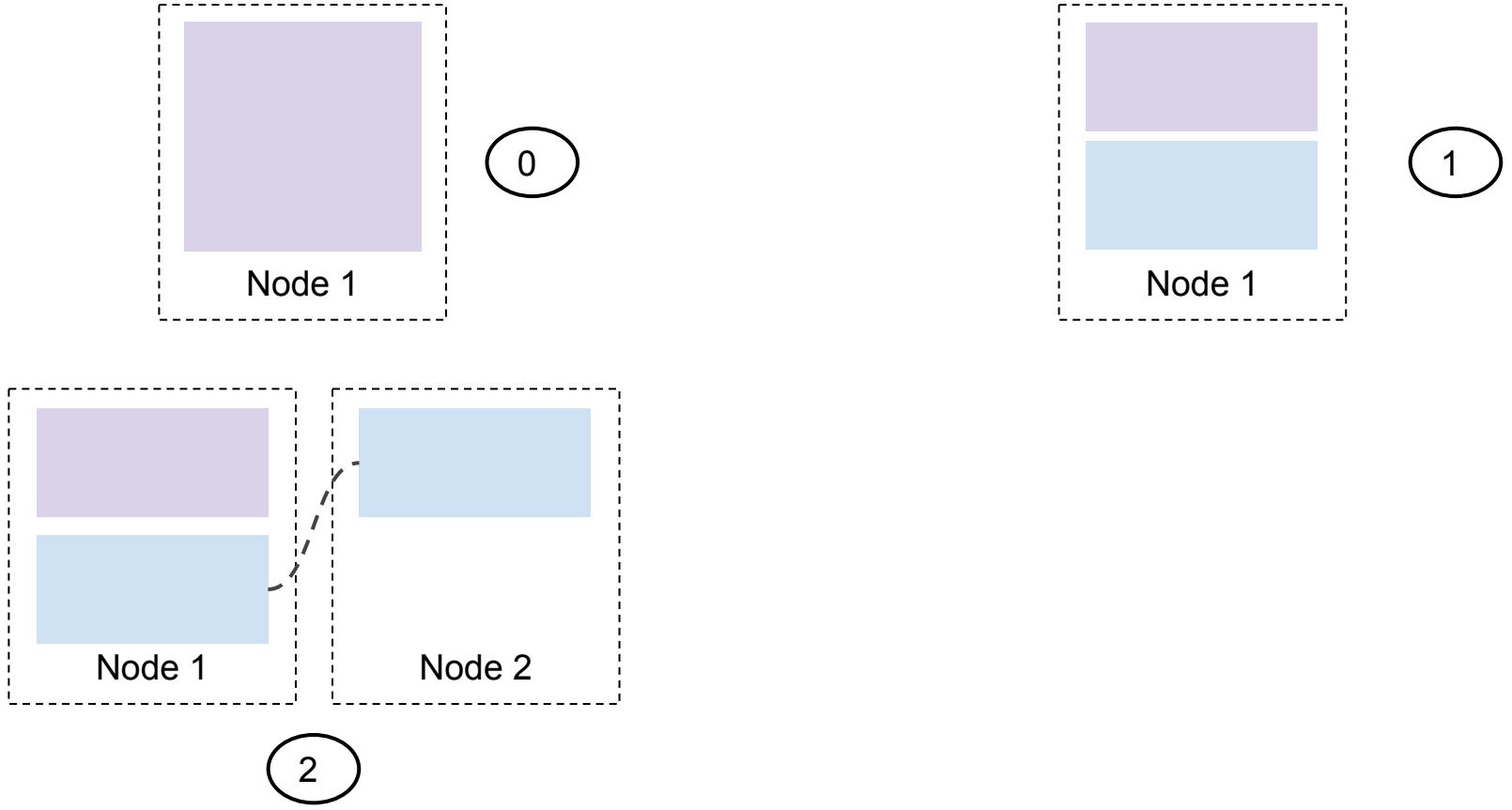


How to scale?

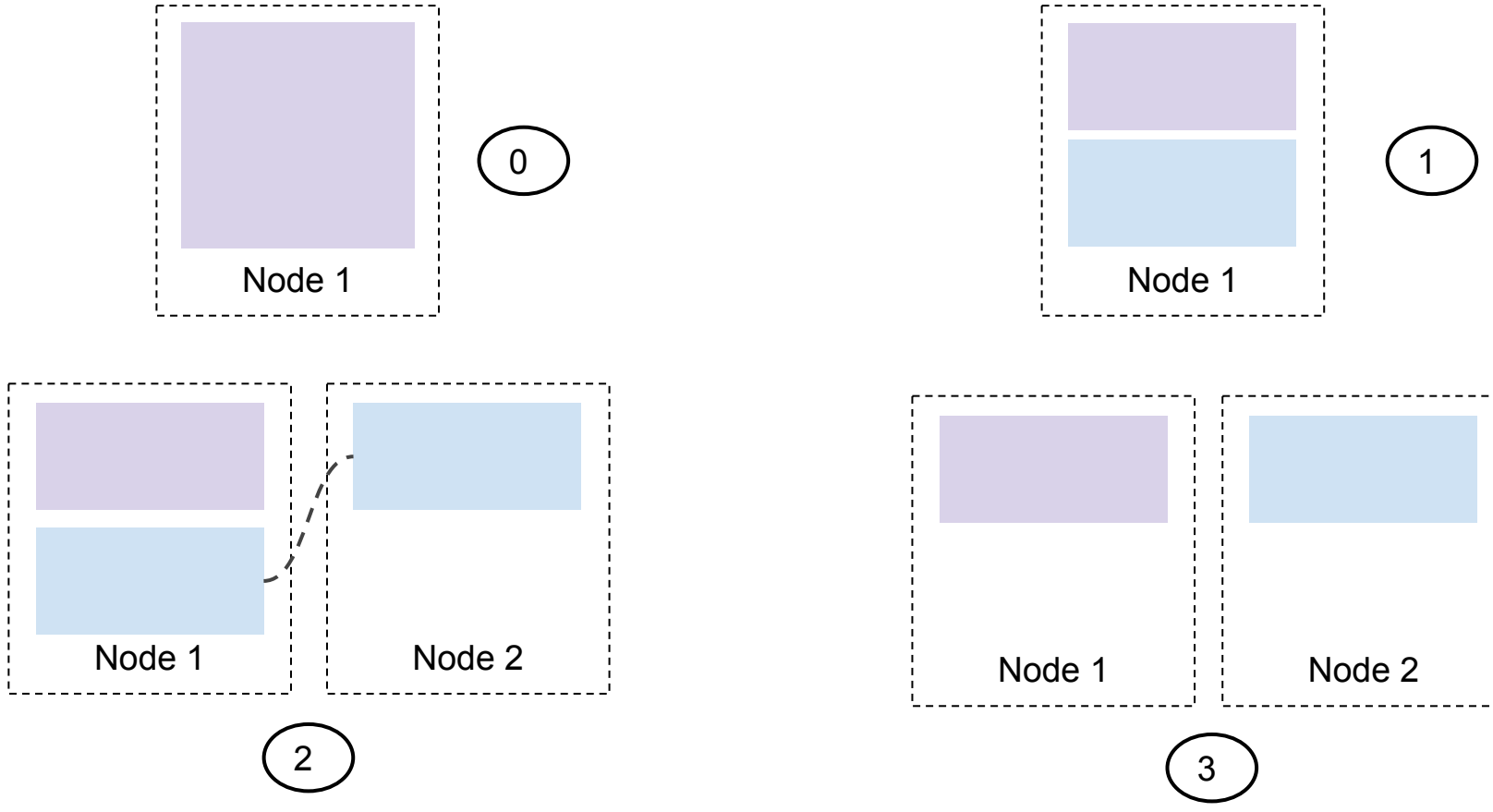
- That's simple
- **Logical split**
- Just Split && Move
- Split safely using Raft



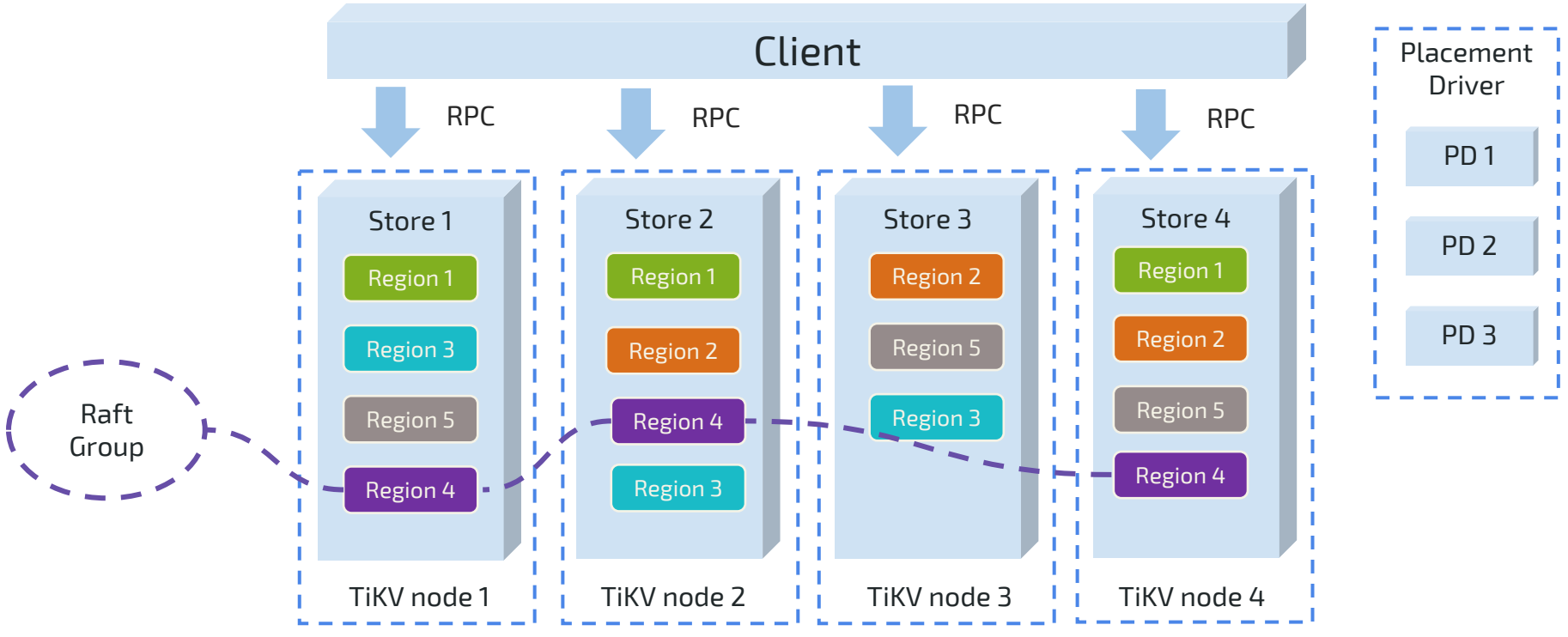
Scale-out (Add new replica in another node)



Scale-out (Remove old replica)



TiKV as a distributed KV engine



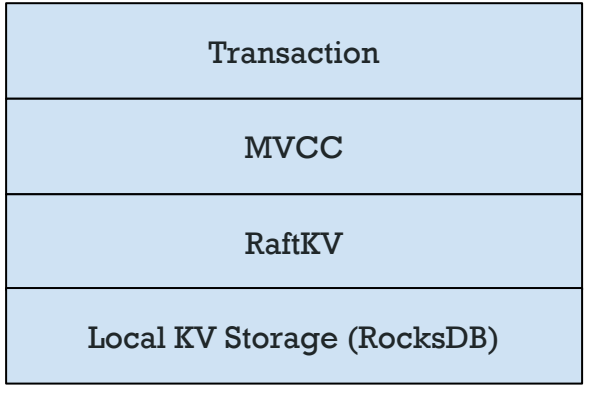
MVCC and Transaction

- MVCC
  - Data layout
    - key1_version2 -> value
    - key1_version1 -> value
    - key2_version3 -> value
  - Lock-free snapshot reads

- Transaction
  - Inspired by Google Percolator
  - 'Almost' decentralized 2-phase commit

TiKV: Architecture overview (Logical)

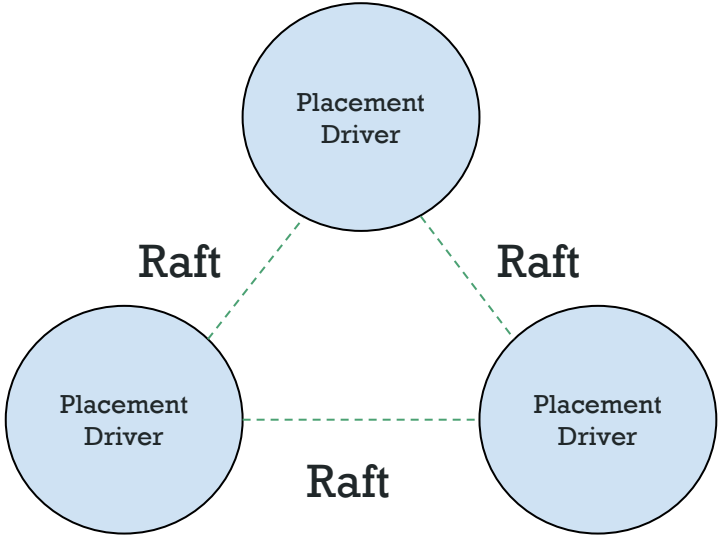
- Highly layered
- Raft for consistency and scalability
- No distributed file system
  - For better performance and lower latency



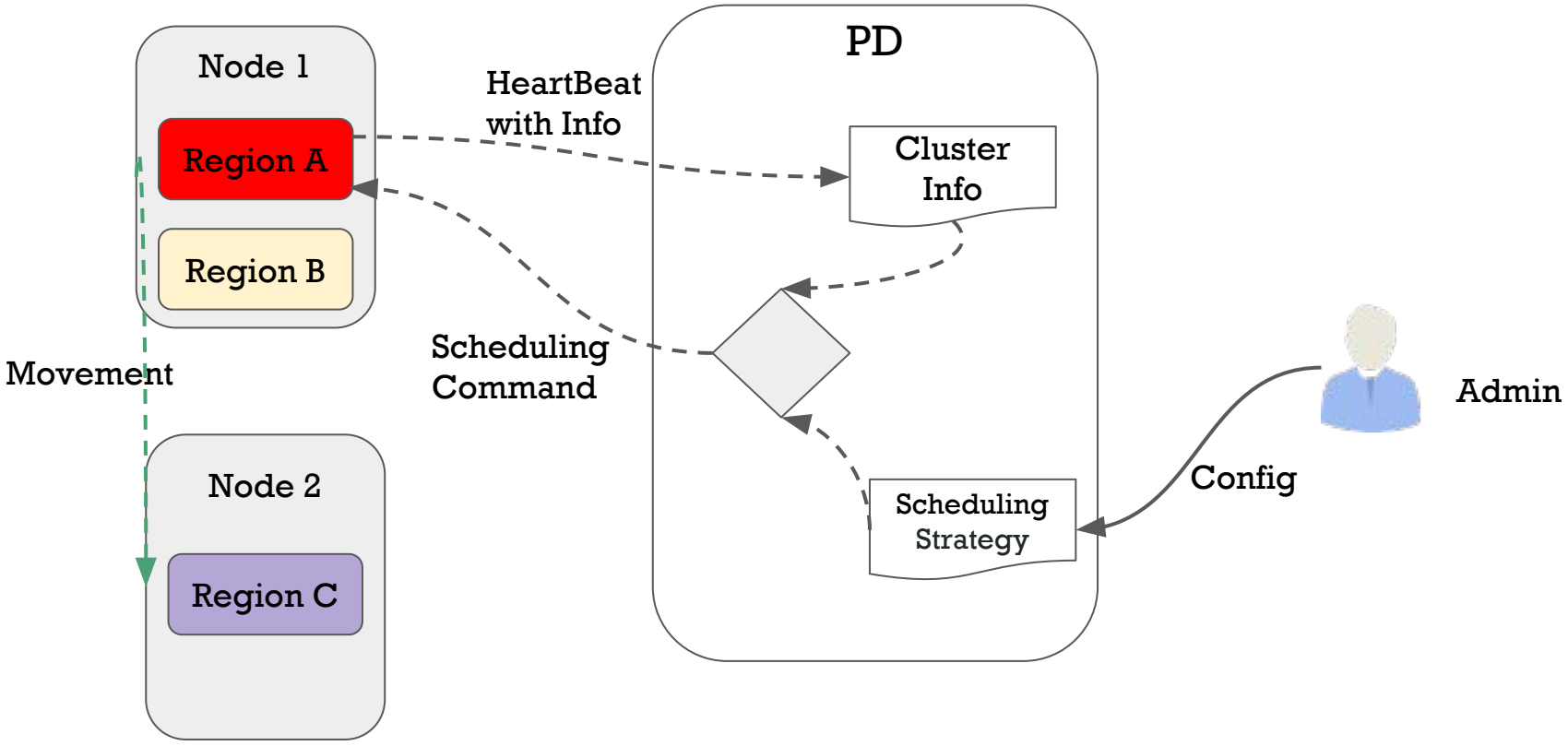
Replica Scheduling

Placement Driver

- Provide the God's view of the entire cluster
- Store the metadata
  - Clients have cache of placement information.
- Maintain the replication constraint
  - 3 replicas, by default
- Data movement for balancing the workload
- It's a cluster too, of course.
  - Thanks to Raft.



PD as the cluster manager



Scheduling Strategy

- Replica number in a raft group
- Replica geo distribution
- Read/Write workload
- Leaders and followers
- Tables and TiKV instances
- Other customized scheduling strategy

TiDB as a SQL database

The SQL Layer

- SQL is simple and very productive
- We want to write code like this:

```
SELECT COUNT(*) FROM user
            WHERE age > 20 and age < 30;
```

The SQL Layer

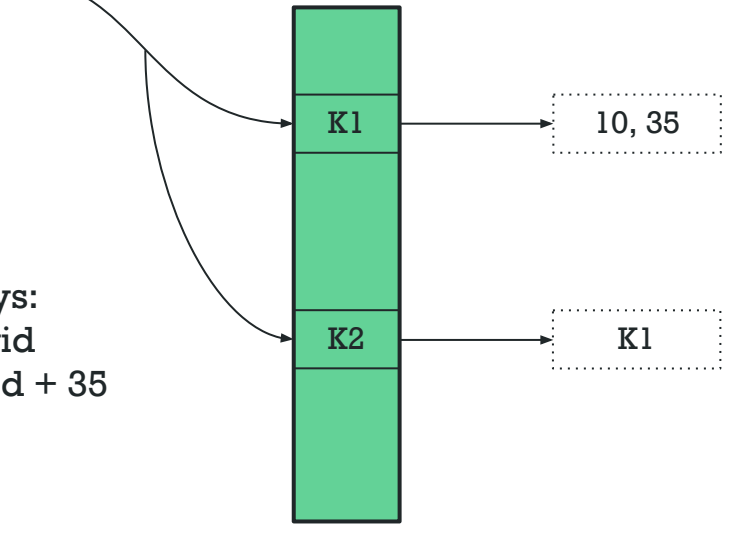
- Mapping relational model to Key-Value model
- Full-featured SQL layer
- Cost-based optimizer (CBO)
- Distributed execution engine

SQL on KV engine

- Row
  - Key: TableID + RowID
  - Value: Row Value
- Index
  - Key: TableID + IndexID + Index-Column-Values
  - Value: RowID

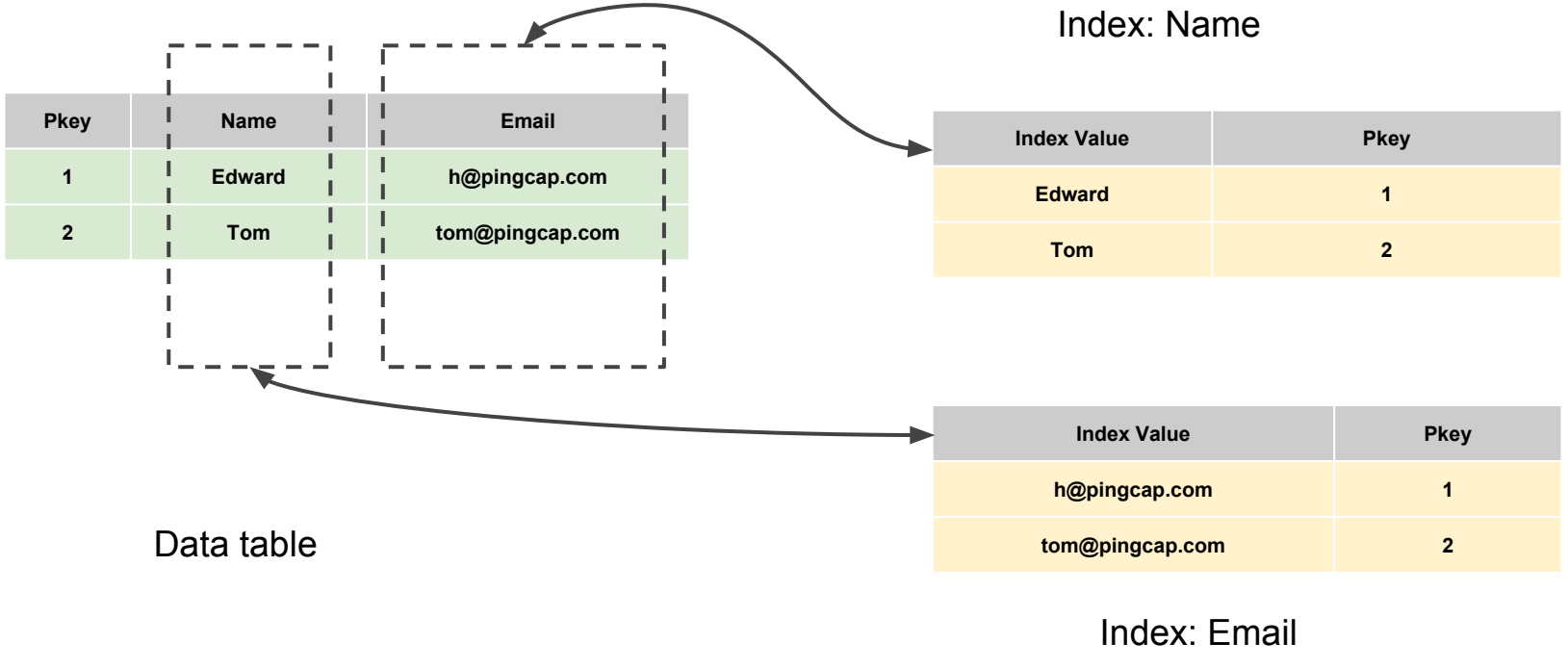
CREATE TABLE `t` (`id` int, `age` int, key `age_idx` (`age`));

INSERT INTO `t` VALUES (100, 35);



Encoded Keys:
K1: tid + rowid
K2: tid + idxid + 35

Secondary Index



SQL on KV engine

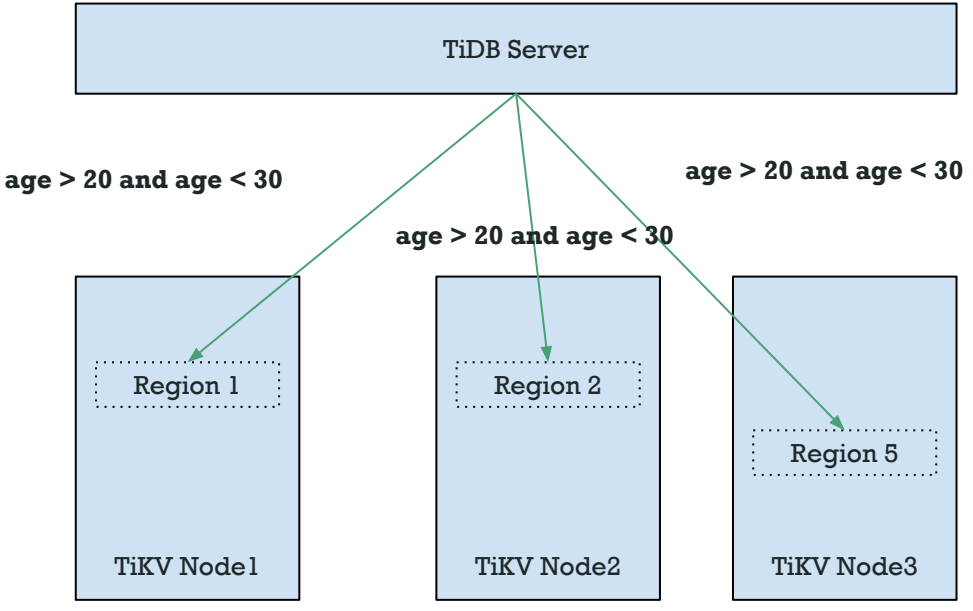
- Key-Value pairs are byte arrays
- Row data and index data are converted into Key-Value
- Key should be encoded using the memory-comparable encoding algorithm
  - compare(a, b) == compare (encode(a), encode(b))
  - Example: Select * from t where age > 10

Index is just not enough...

- Can we push down filters?
  - `select count(*) from person where age > 20 and age < 30`
- It should be much faster, maybe 100x
  - Less RPC round trip
  - Less transferring data

Distributed Execution Engine

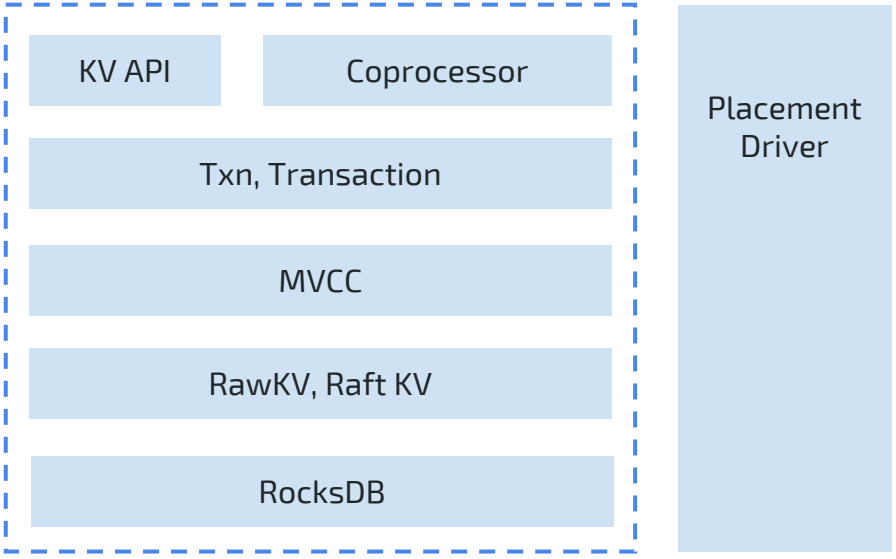
TiDB knows that Region 1 / 2 / 5 stores the data of person table.

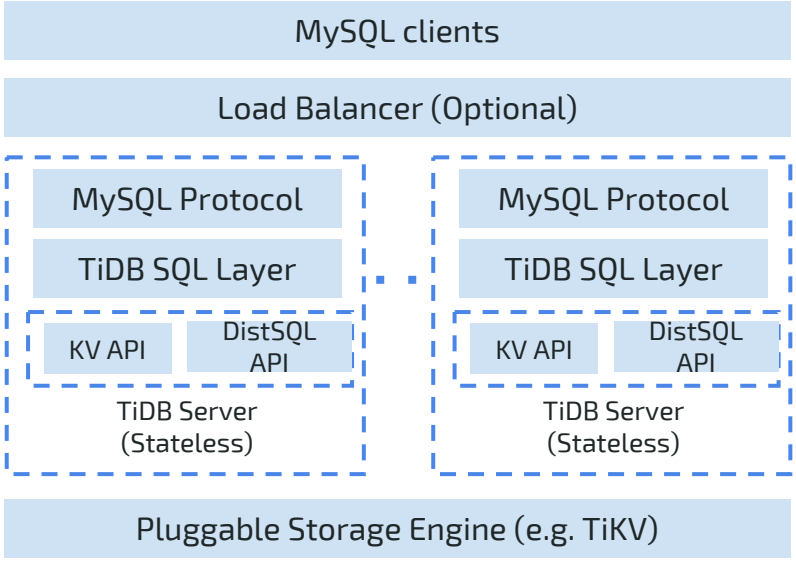


What about drivers for every language?

- We just build a protocol layer that is compatible with MySQL. Then we have all the MySQL drivers.
  - All the tools
  - All the ORMs
  - All the applications
- That's what TiDB does.

Architecture

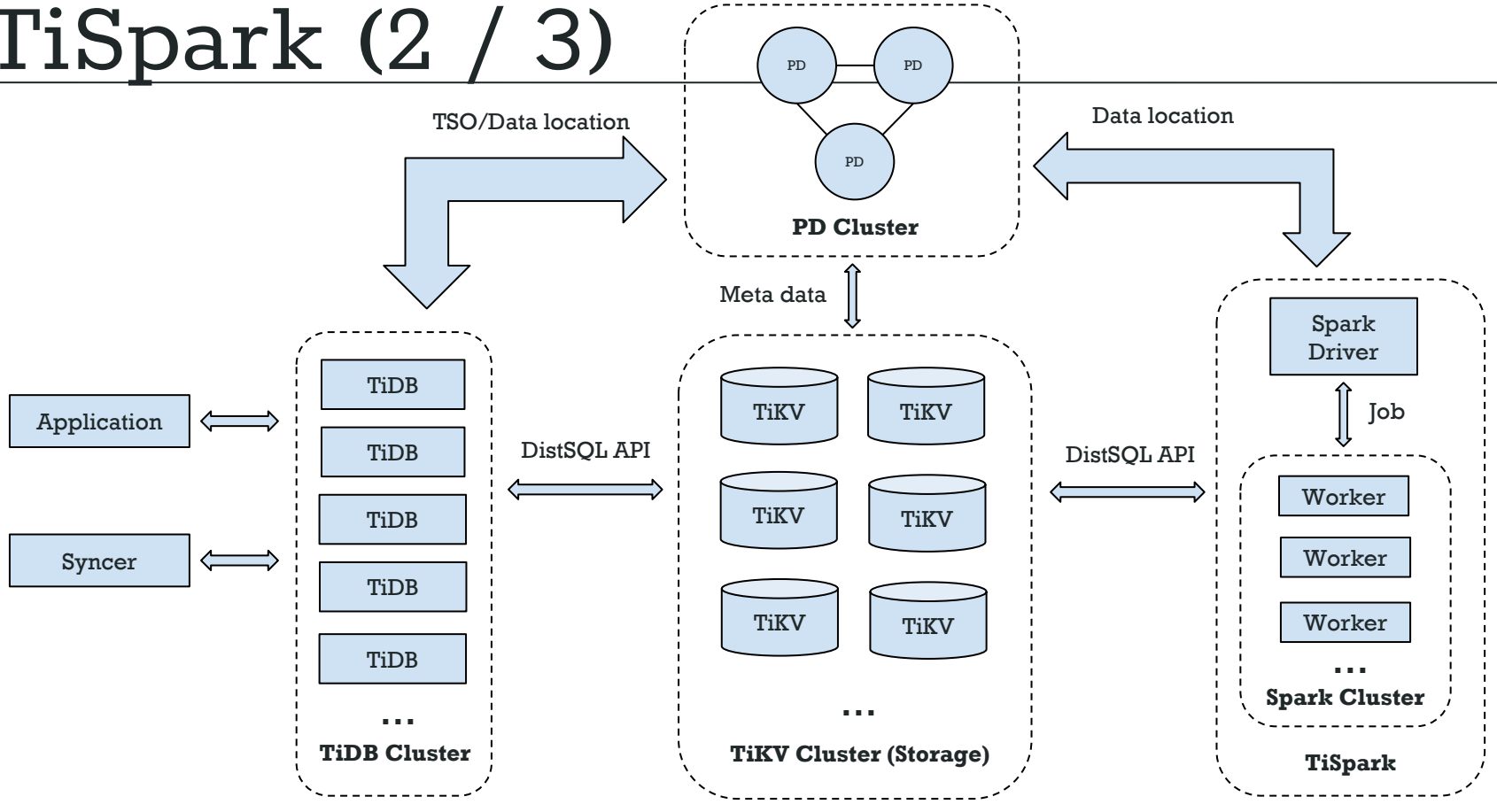




TiSpark (1 / 3)

- TiSpark = SparSQL on TiKV
  - SparkSQL directly on top of a distributed Database Storage
- Hybrid Transactional/Analytical Processing(HTAP) rocks
  - Provide strong OLAP capacity together with TiDB
- Spark ecosystem

TiSpark (2 / 3)


PD
PD
PD
PD Cluster
TSO/Data location
Data location
Meta data
Application
Syncer
TiDB
TiDB
TiDB
TiDB
TiDB
...
TiDB Cluster
DistSQL API
TiKV
TiKV
TiKV
TiKV
TiKV
TiKV
...
TiKV Cluster (Storage)
DistSQL API
Spark Driver
Job
Worker
Worker
Worker
...
Spark Cluster
TiSpark

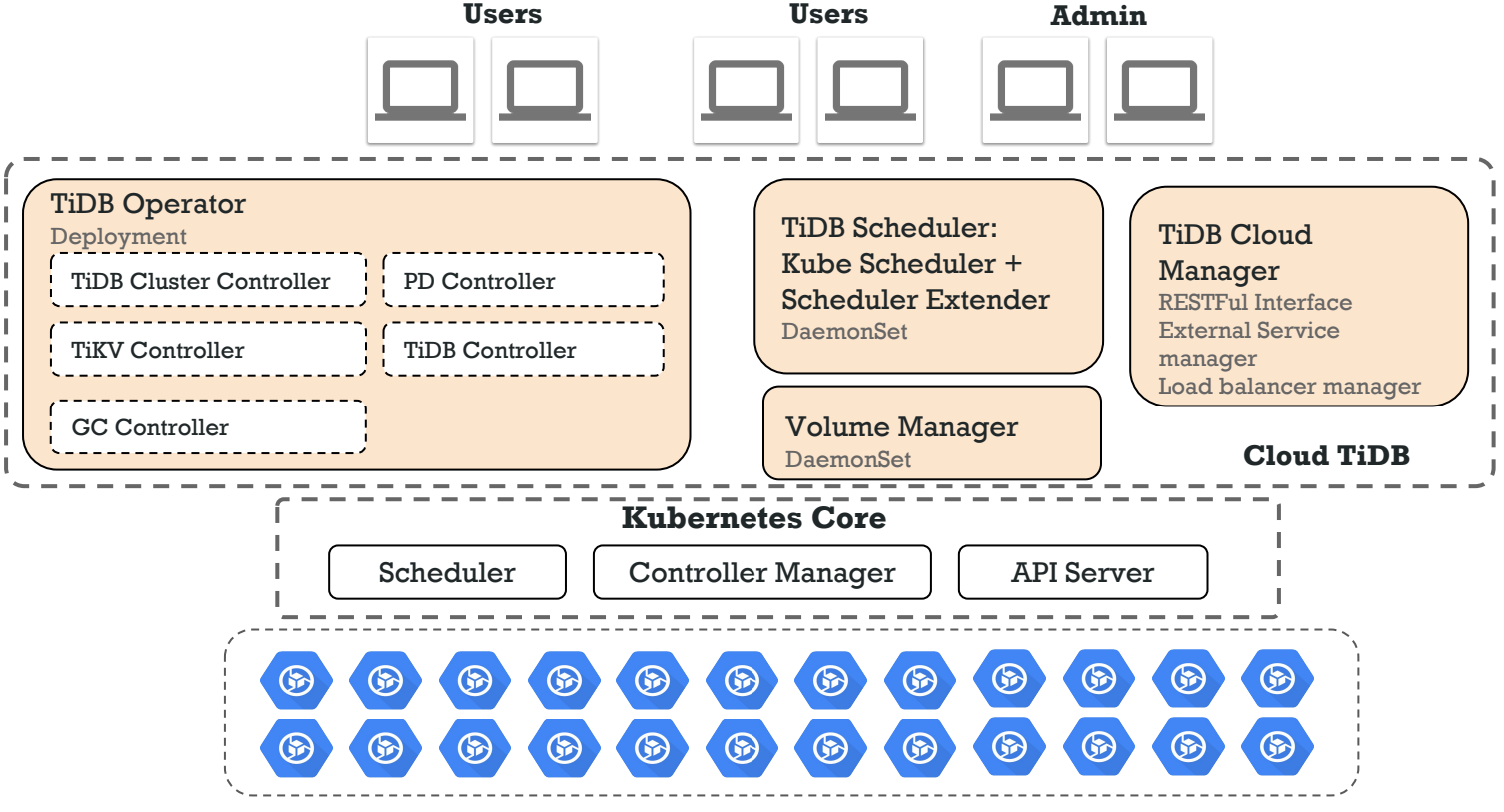
TiSpark (3 / 3)

- TiKV Connector is better than JDBC connector
- Index support
- Complex Calculation Pushdown
- CBO
  - Pick up the right Access Path
  - Join Reorder
- Priority & Isolation Level

TiDB as a Cloud-Native Database

Deploy a database on the cloud

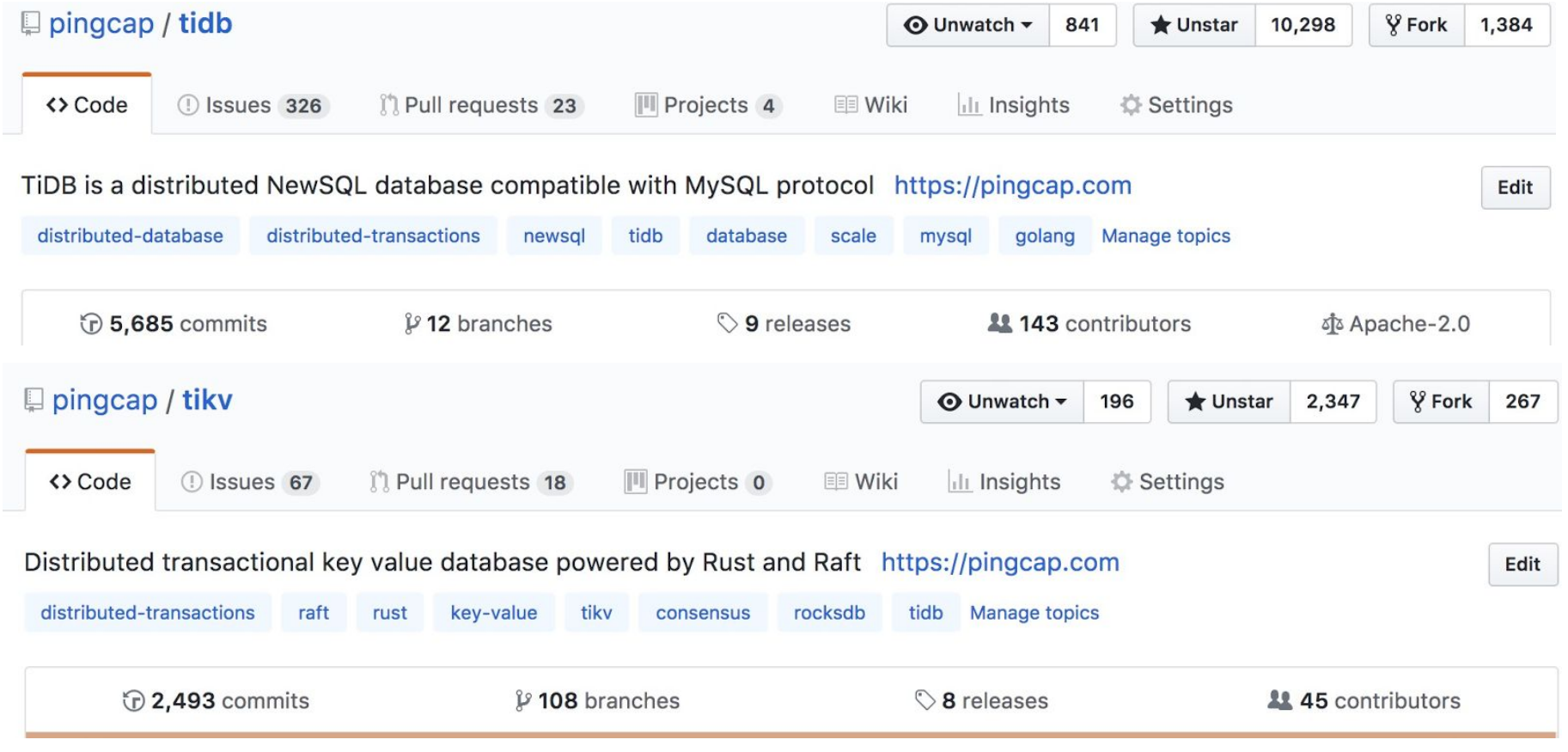
TiDB on Kubernetes



Cloud TiDB

Open Source

Open Source



pingcap / tidb

Unwatch 841 | Unstar 10,298 | Fork 1,384

Code | Issues 326 | Pull requests 23 | Projects 4 | Wiki | Insights | Settings

TiDB is a distributed NewSQL database compatible with MySQL protocol https://pingcap.com

Edit

distributed-database distributed-transactions newsql tidb database scale mysql golang Manage topics

5,685 commits | 12 branches | 9 releases | 143 contributors | Apache-2.0

pingcap / tikv

Unwatch 196 | Unstar 2,347 | Fork 267

Code | Issues 67 | Pull requests 18 | Projects 0 | Wiki | Insights | Settings

Distributed transactional key value database powered by Rust and Raft https://pingcap.com

Edit

distributed-transactions raft rust key-value tikv consensus rocksdb tidb Manage topics

2,493 commits | 108 branches | 8 releases | 45 contributors

Roadmap

- Multi-tenant
- Better Optimizer and Runtime
- Performance Improvement
- Document Store
- Backup & Reload & Migration Tools

Thanks

Q&A

https://github.com/pingcap/tidb

https://github.com/pingcap/tikv

https://github.com/pingcap/pd

https://github.com/pingcap/tispark

https://github.com/pingcap/docs

https://github.com/pingcap/docs-cn

Contact Me: sunhao@pingcap.com