Scale out your database without pain

Liu Qi

About me

Qi Liu (刘奇)

Co-founder & CEO of PingCAP

JD / Wandou Labs / PingCAP

Infrastructure software engineer / Open source hacker

Codis / TiDB / TiKV

Let's get started

Thanks to statelessness, we can easily scale **LA P(**without **M)** by adding more machines

Hold on, how about "**M**" ? (MySQL)

What would you do when…

RDBMS is becoming the **performance bottleneck** of your backend service

The **amount** of data stored in RDBMS is overwhelming

What would you do when…

RDBMS is becoming the performance bottleneck of your backend service

The amount of data stored in RDBMS is overwhelming

Sharding your database or table **manually**, yeah

  Or

Sharding by using proxy (vitess, MySQL proxy, kingshard, atlas…)

When things get more complicated…

You want to do some complex queries on a sharding cluster

e.g. simple JOIN or GROUP BY or Subquery

Oh, No...

You need a real distributed database, not sharding.

TiDB: when you need a distributed SQL database

Horizontal scalability (forget about sharding key)

Asynchronous schema changes(feel free to change your schema online)

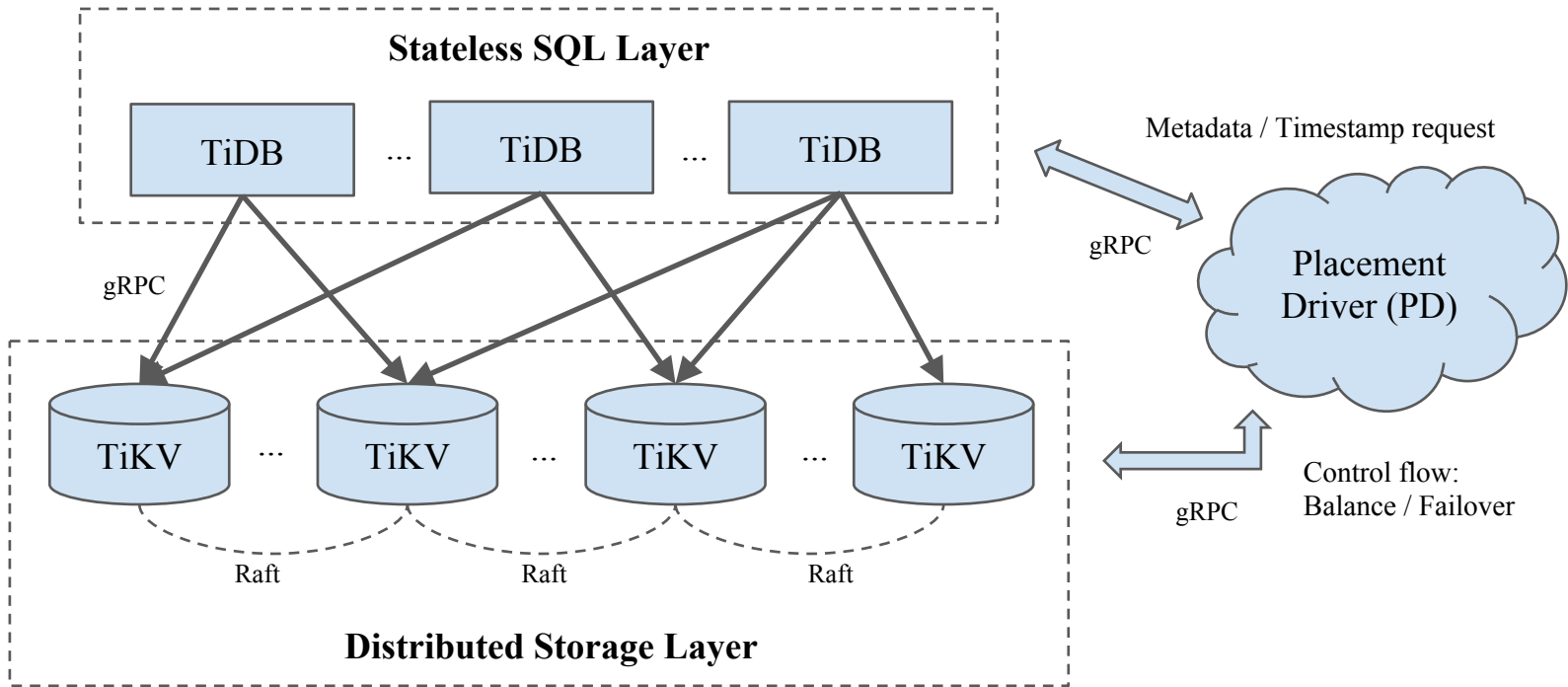Consistent distributed transactions (no sharding limits anymore)

Compatible with the MySQL protocol (You know what I mean)

MySQL Sharding VS TiDB

	MySQL Sharding	TiDB
ACID Transaction Support	✖	✔
Elastic Scaling	✖	✔
Complex Query	✖	✔
Failover	Manual	Auto
MySQL Compatibility	Low	High
Max Capacity (high performance)	< 2TB	200 TB+

Architecture

Stateless SQL Layer

TiDB ... TiDB ... TiDB

gRPC

Metadata / Timestamp request

gRPC

Placement Driver (PD)

TiKV ... TiKV ... TiKV ... TiKV

Raft   Raft   Raft

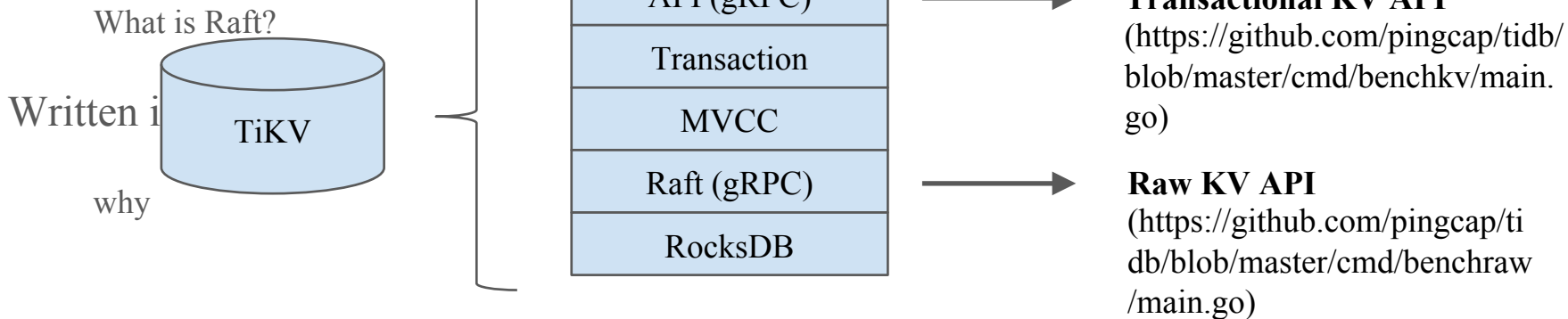Control flow: Balance / Failover

gRPC

Distributed Storage Layer

Storage stack 1/2

TiKV is the underlying storage layer

Physically, data is stored in RocksDB

We build a Raft layer on top of RocksDB

What is Raft?

Written i~~~TiKV~~~

why

API (gRPC)
Transaction
MVCC
Raft (gRPC)
RocksDB

Transactional KV API
(https://github.com/pingcap/tidb/blob/master/cmd/benchkv/main.go)

Raw KV API
(https://github.com/pingcap/tidb/blob/master/cmd/benchraw/main.go)

Storage stack 2/2

Data is organized by **Regions**

Region: a set of continuous key-value pairs

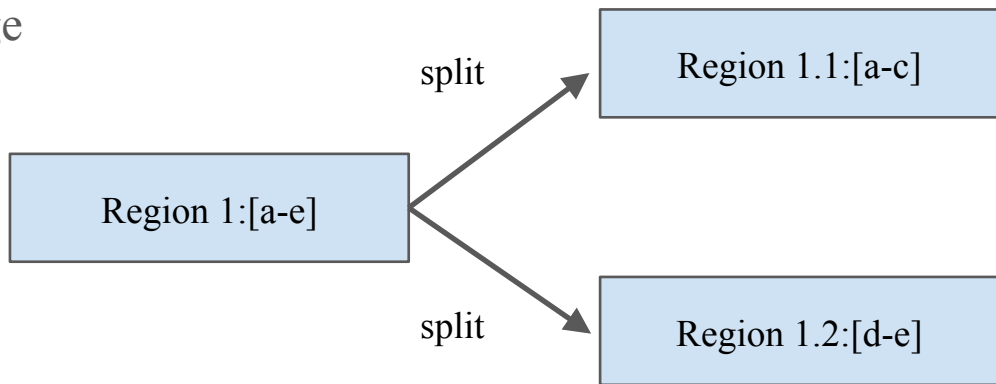| RPC (gRPC) |
| Transaction |
| MVCC |
| Raft |
| RocksDB |

…...
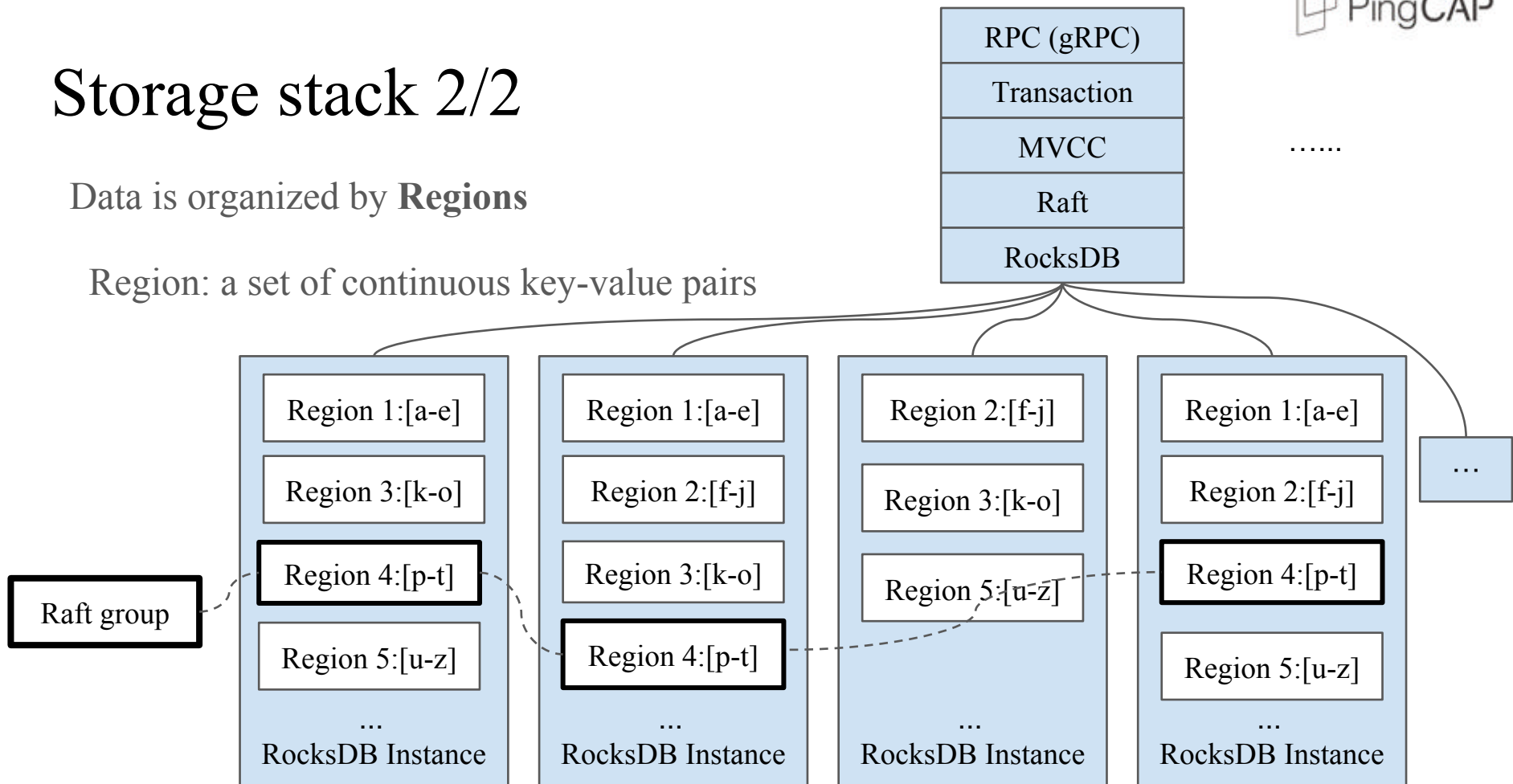
Region size

Spanner: ~1G

TiDB: 64M ~ 1G   (configurable)

Depending on the network quality

Dynamic Multi-Raft

What's Dynamic Multi-Raft?

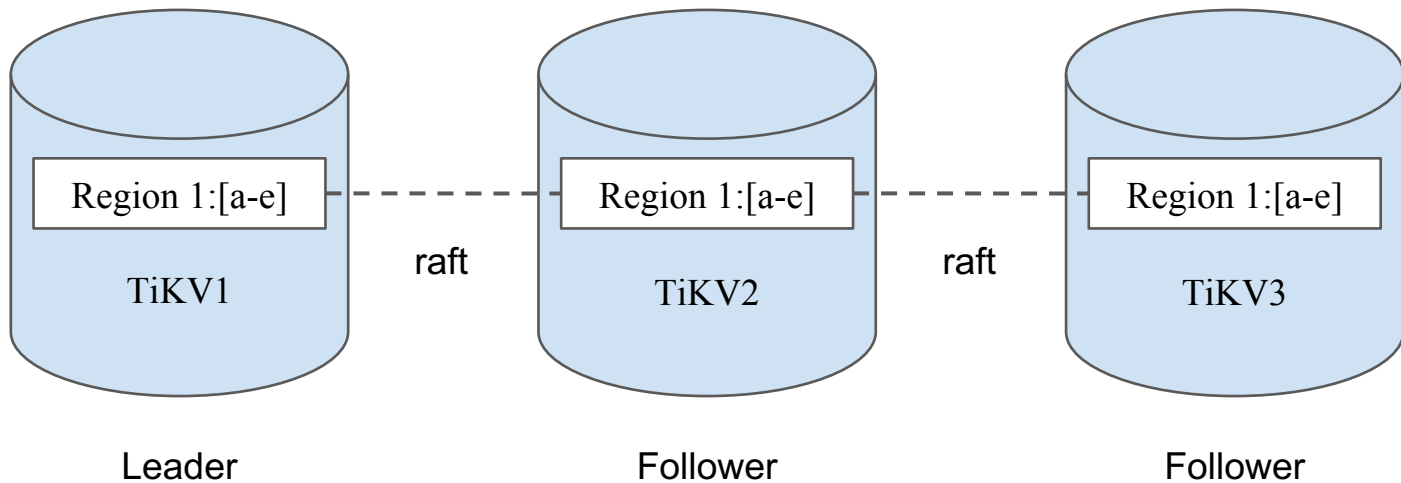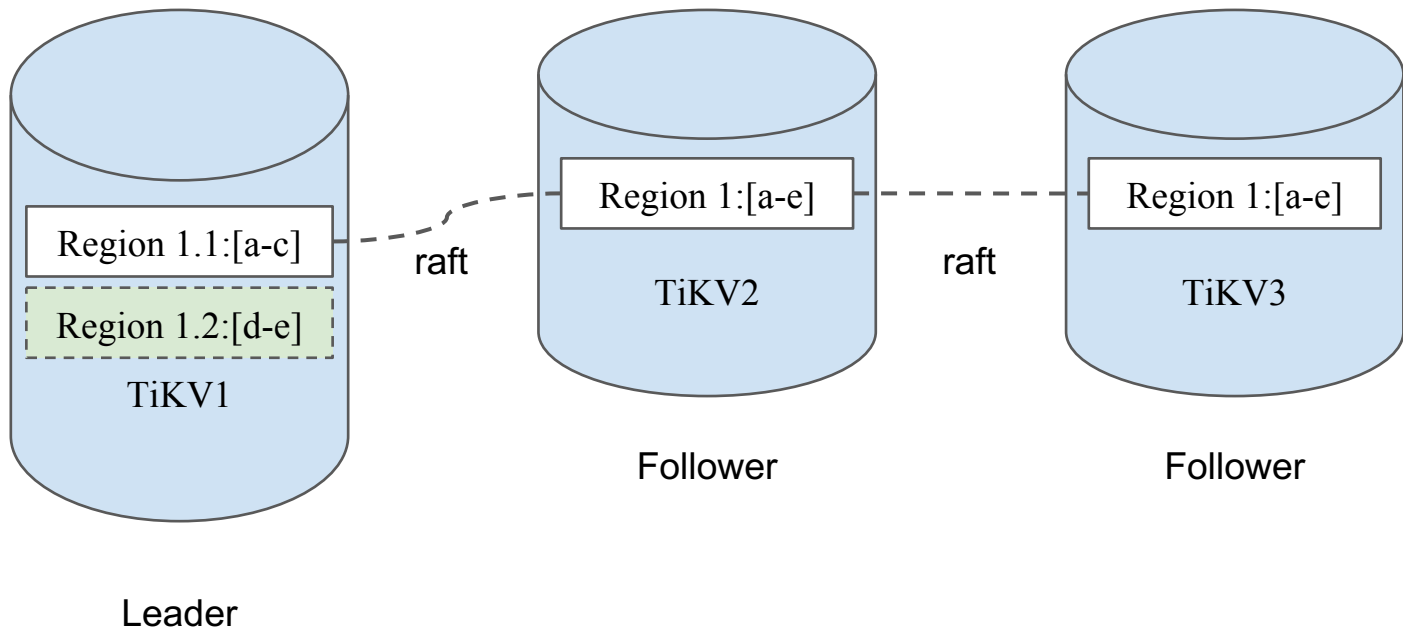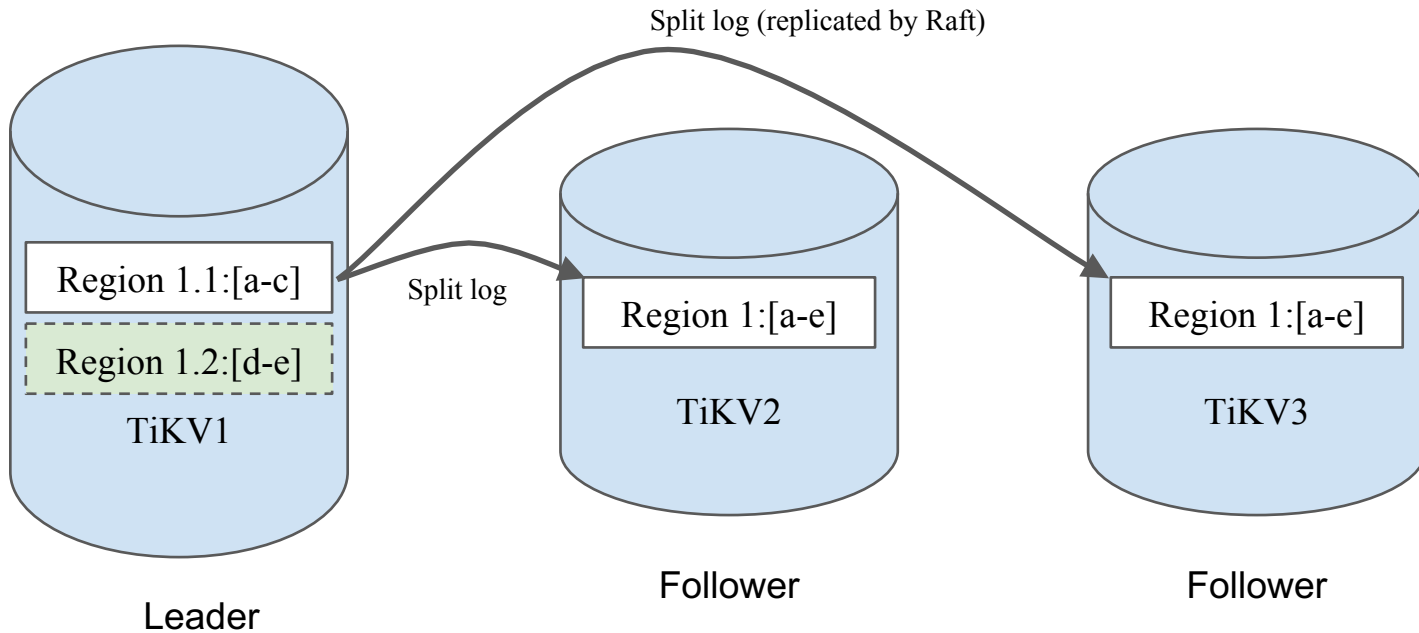Dynamic split / merge

Safe split / merge

```
                                        ┌──────────────────────┐
                             split      │  Region 1.1:[a-c]     │
                          ┌────────────▶│                      │
                          │             └──────────────────────┘
┌──────────────────────┐  │
│  Region 1:[a-e]      │──┤
│                      │  │
└──────────────────────┘  │             ┌──────────────────────┐
                             split       │  Region 1.2:[d-e]    │
                          └────────────▶│                      │
                                        └──────────────────────┘
```

Safe Split: 1/4

Raft group

Region 1:[a-e]

TiKV1

raft

Region 1:[a-e]

TiKV2

raft

Region 1:[a-e]

TiKV3

Leader

Follower

Follower

Safe Split: 2/4

Region 1.1:[a-c]

Region 1.2:[d-e]

TiKV1

raft

Region 1:[a-e]

TiKV2

raft

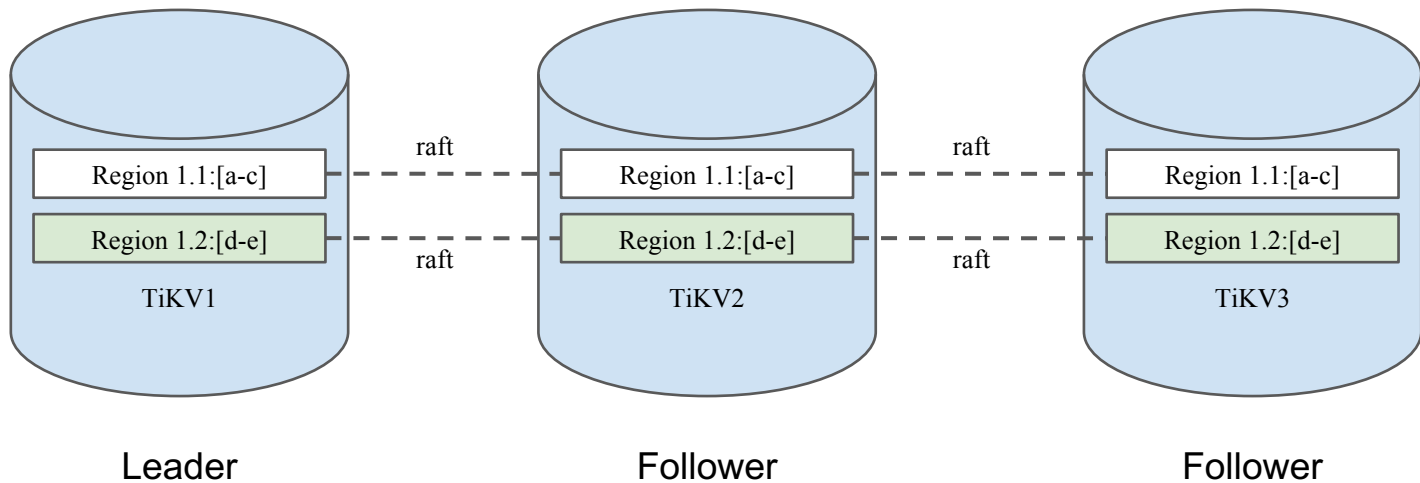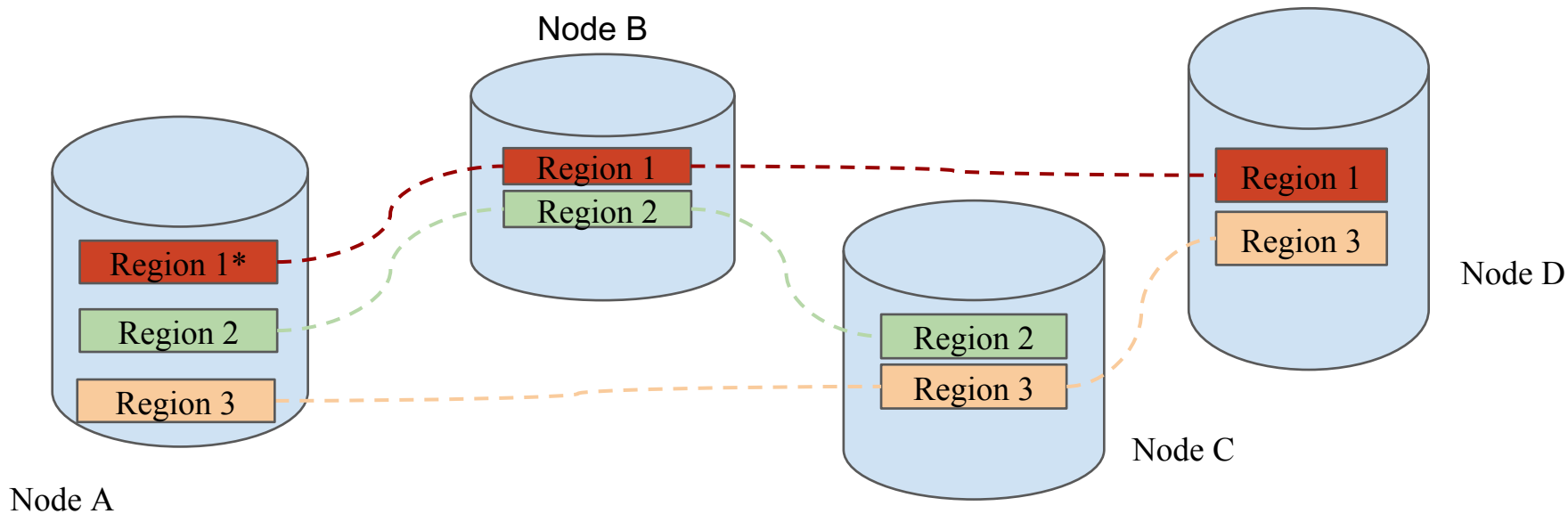Region 1:[a-e]

TiKV3

Leader

Follower

Follower

Safe Split: 3/4

Safe Split: 4/4

Scale-out (initial state)
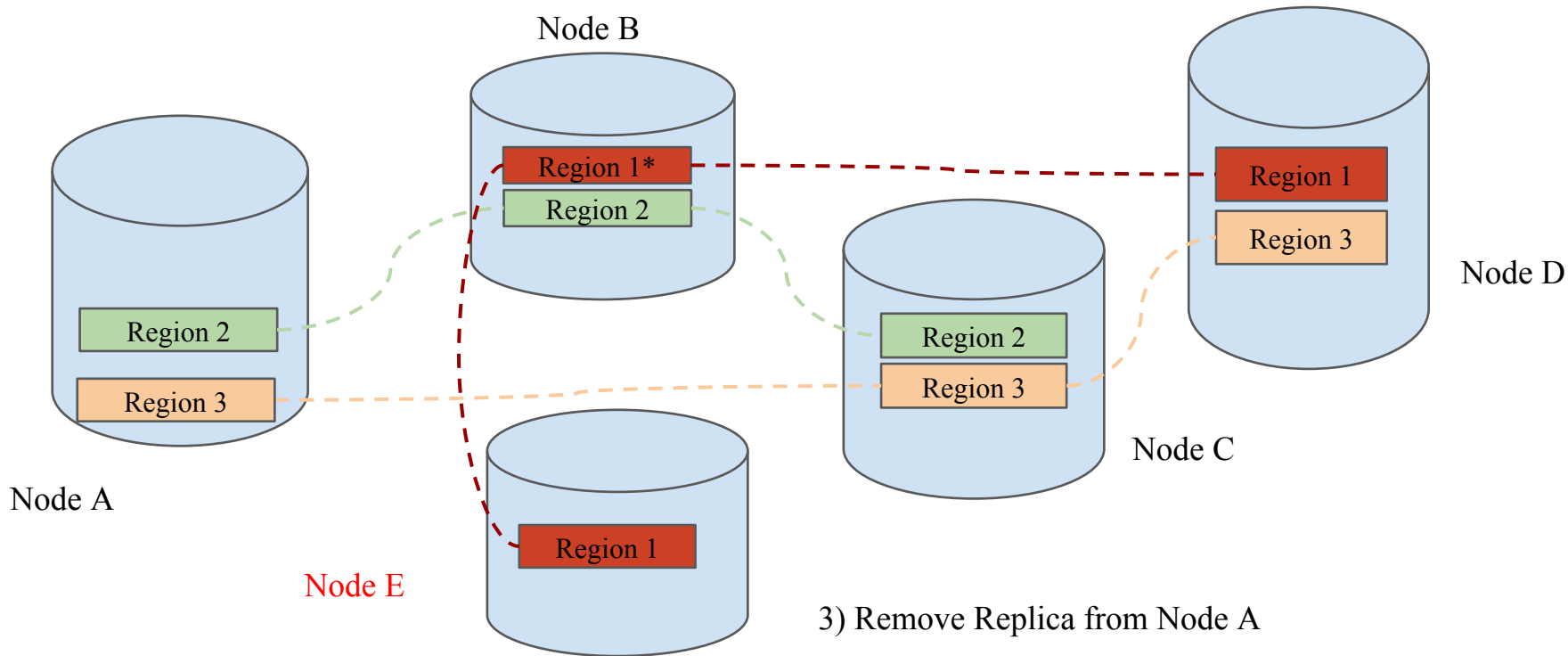
Scale-out (add new node)

Node B

Node A

Node C

Node D

Node E

Region 1*

Region 2

Region 3

Region 1^

Region 2

Region 2

Region 3

Region 1

Region 3

1) Transfer leadership of region 1 from Node A to Node B

Scale-out (balancing)

Node B

Node A

Node C

Node D

Node E

2) Add Replica on Node E

Region 1* | Region 2 | Region 1 | Region 3 | Region 1 | Region 2 | Region 3 | Region 2 | Region 3

Scale-out (balancing)

Node B

Node A

Node E

Node C

Node D

Region 1* · Region 1
Region 2
Region 2
Region 3
Region 3
Region 1
Region 2
Region 3
Region 1

3) Remove Replica from Node A

Placement Driver

The concept comes from Spanner

Provide the God's view of the entire cluster

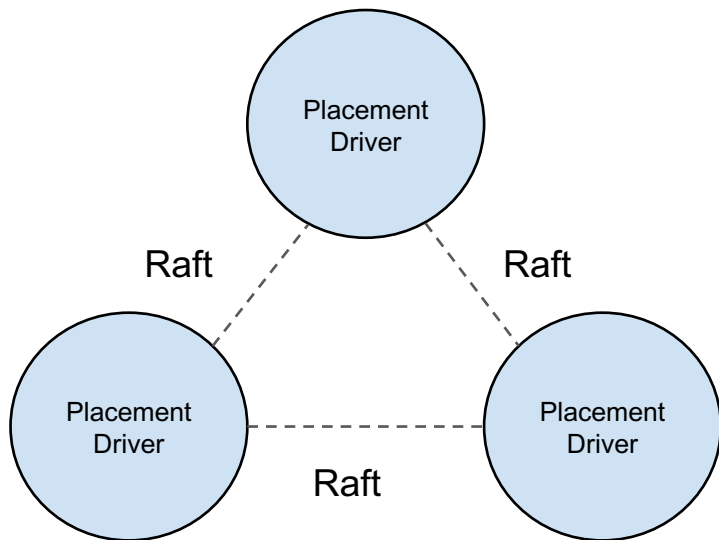Store the metadata

    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.

ACID Transaction

The transaction API is in TiKV

Based on Google Percolator

'Almost' decentralized 2-phase commit

Timestamp Allocator (PD)

~8M timestamps allocations per second

Optimistic transaction model

Default isolation level: Repeatable Read

External consistency: Snapshot Isolation + Lock
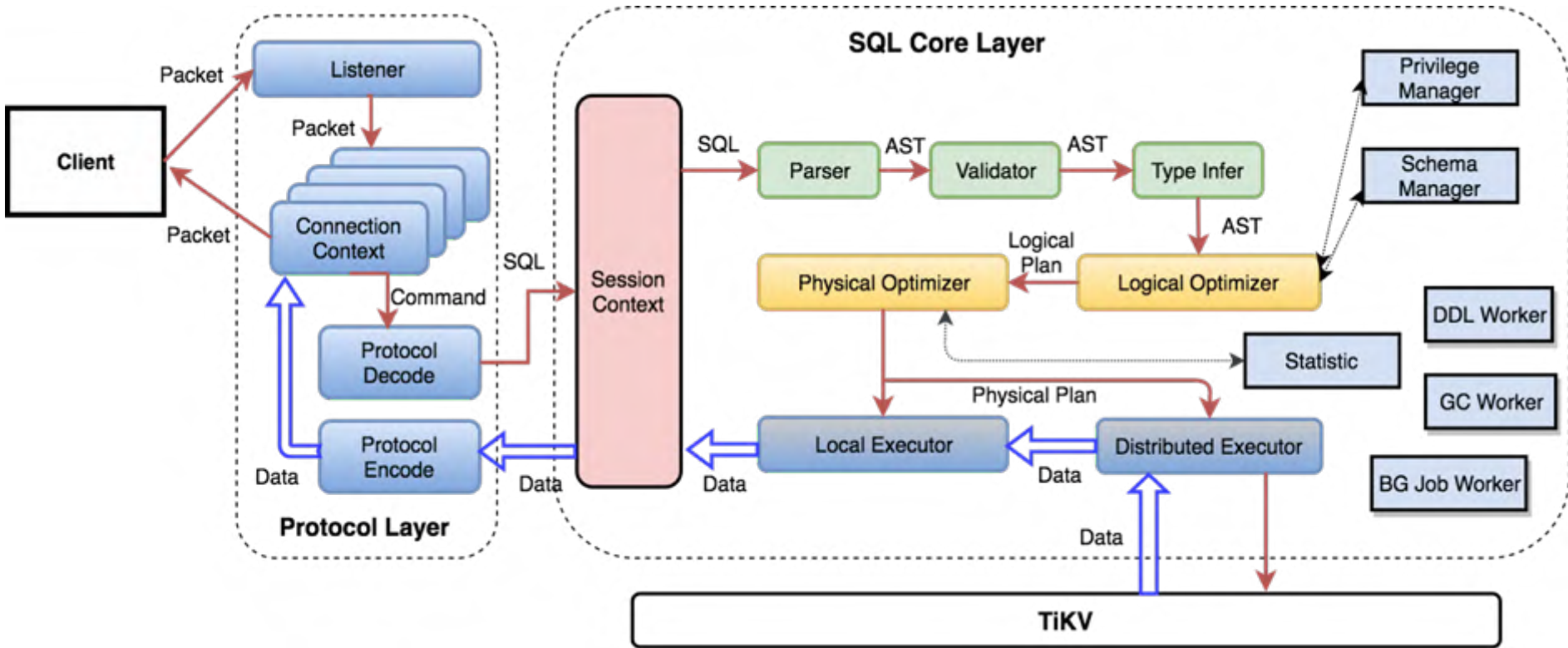
Distributed SQL

Full-featured SQL layer

Predicate pushdown

Distributed join

Distributed cost-based optimizer (Distributed CBO)
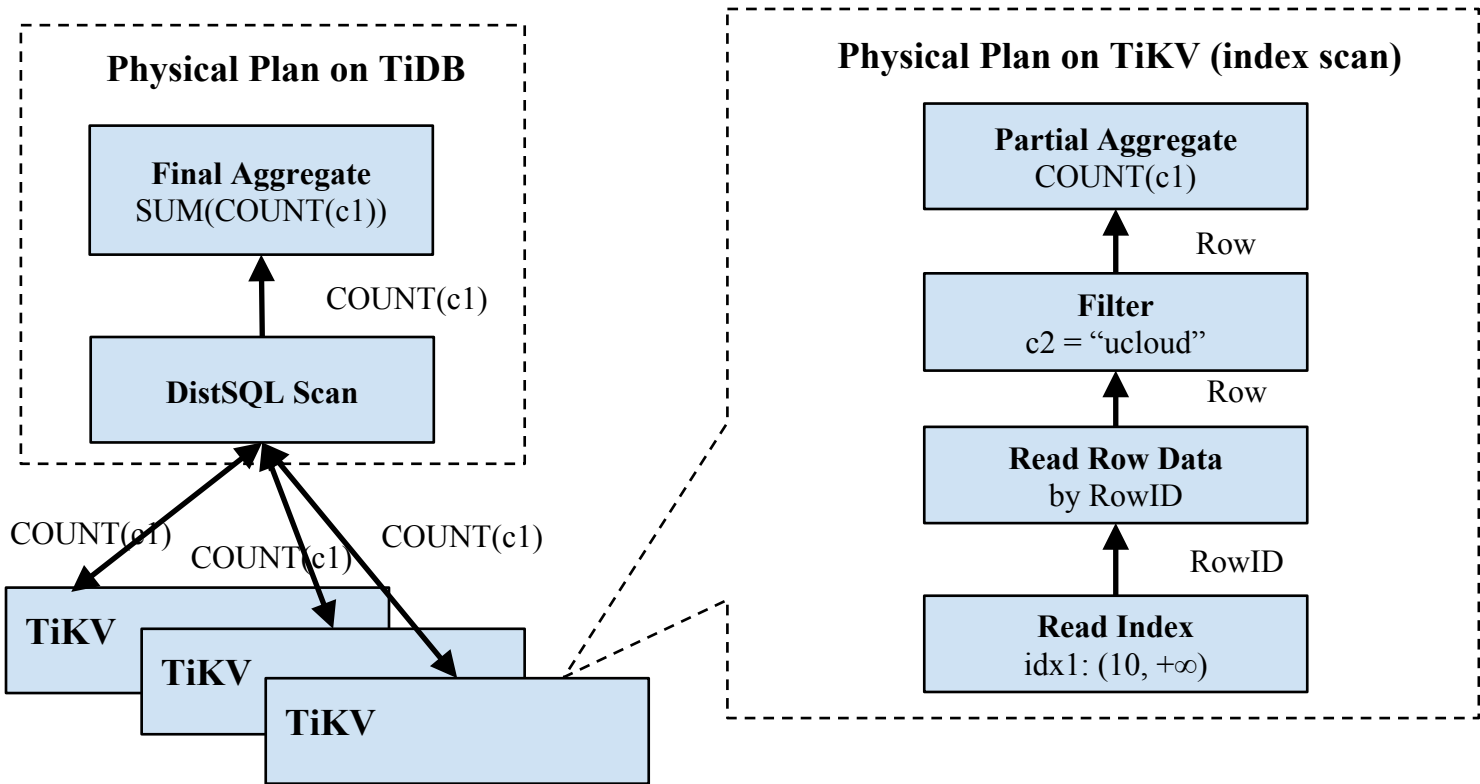
TiDB SQL Layer overview

What happens behind a query

CREATE TABLE t (c1 INT, c2 TEXT, KEY idx_c1(c1));

SELECT COUNT(c1) FROM t WHERE c1 > 10 AND c2 = 'foo';

Query Plan

Physical Plan on TiDB

Final Aggregate
SUM(COUNT(c1))

COUNT(c1)

DistSQL Scan

COUNT(c1)   COUNT(c1)   COUNT(c1)

TiKV

TiKV

TiKV

Physical Plan on TiKV (index scan)

Partial Aggregate
COUNT(c1)

Row

Filter
c2 = "ucloud"

Row

Read Row Data
by RowID

RowID

Read Index
idx1: (10, +∞)

Supported Distributed Join Type

Hash Join

Sort merge Join

Index-lookup Join

No silver bullet (anti-patterns for TiDB SQL)

Join between large tables without index or any hints

Get distinct values from large tables without index

Sort without index

Result set is too large (forget LIMIT N?)

What makes TiDB slow?

Hot and **small** table accessing

> TiDB can't redistribute your workload

If you always **append** to 1 table, the hotspot will always exist in the last region of this table.

> TiDB isn't a time series database

Auto-increment ID / Timestamp index

Best practices

Don't use distributed database if you can handle your data with a **single** MySQL
instance

Best practices

Use [ansible](#) to maintain your cluster (for private deployment)

Don't use network block devices, please use **ssd**

Careful about the Auto Increment ID (always append to a table)

Best practices

Read the documents first

Need help?

File issues

Google Group

Email: info@pingcap.com

Tools matter

Syncer

TiDB-Binlog

Mydumper/MyLoader(loader)

https://github.com/pingcap/tidb-tools
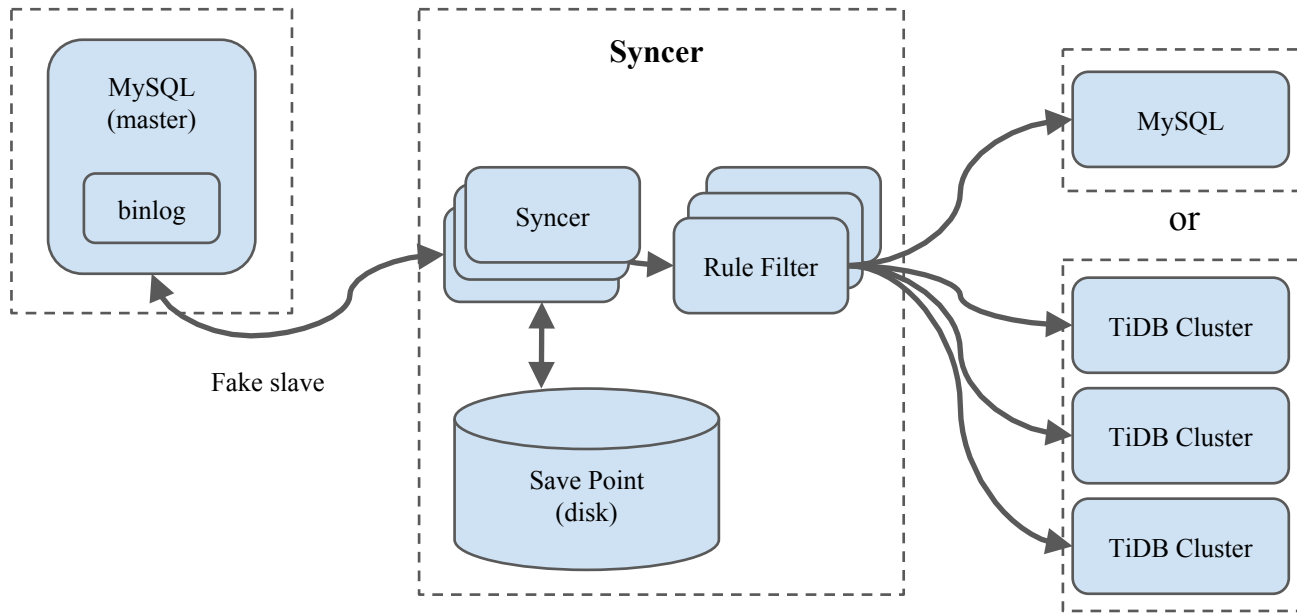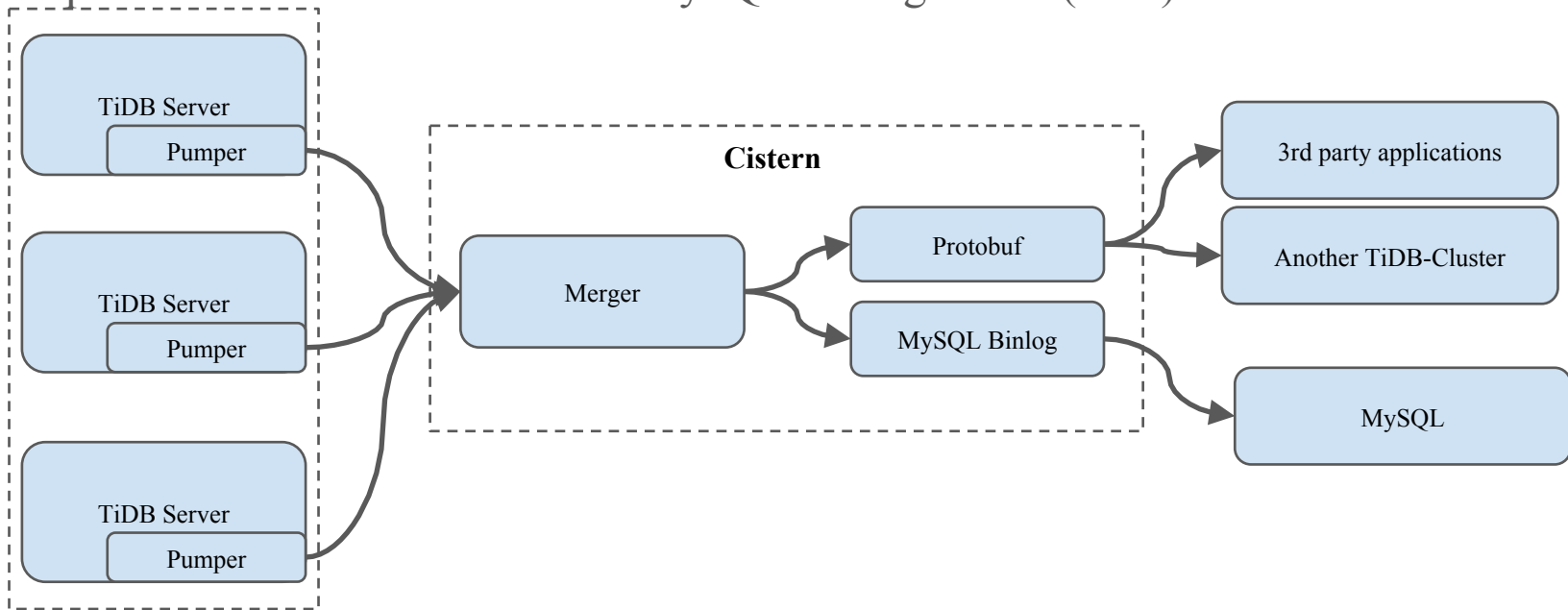
Syncer

Synchronize data from MySQL in real-time

Hook up as a MySQL replica

TiDB-Binlog

Subscribe the incremental data from TiDB

Output Protobuf formatted data or MySQL Binlog format(WIP)

PingCAP

Roadmap

TiSpark: Integrate TiKV with SparkSQL （2017.07）

Better optimizer (Statistic && CBO)

Json type and **document store** for TiDB （2017.09）

> MySQL 5.7.12+ X-Plugin

Integrate with Kubernetes （2017.07）

Integrate with UCloud and more, yay~

Thank you

https://github.com/pingcap/tidb

https://github.com/pingcap/tikv

Any questions?