

01 NewSQL简介

1. 什么是NewSQL
   1. **SQL** 的问题
   2. **noSQL** 的优势与不⾜
   3. **newSQL** 特性
2. CAP定理
3. TiDB简介
   1. TiDB整体概述
   2. TiKV存储引擎简介
   3. TiDB-SQL引擎
   4. TiDB-HTAP介绍
4. 什么是**NewSQL**

数据库发展⾄今有3代了：

1. SQL，传统关系型数据库，例如 MySQL,PG,oracle
2. noSQL，例如 MongoDB,Redis,hbase 3.newSQL:TiDB，OB。。。
   1. **SQL** 的问题

互联⽹在本世纪初开始迅速发展，互联⽹应⽤的⽤户规模、数据量都越来越⼤，并且要求

7X24⼩时在线。

传统关系型数据库在 种环境下成为了瓶颈，通常有2种解决⽅法：

* 升级服务器硬件，提升了性能，但总有天花板。
* 数据分⽚，使⽤分布式集群结构，对单点数据库进⾏数据分⽚，存放到由廉价机器组成的分布式的集群⾥。缺点:1.需要使⽤数据库分⽚中间件做跨库，跨库join、跨库事务难度很⼤.

2.⾮云原⽣

* 1. **noSQL** 的优势与不⾜

noSQL 出现了，放弃了传统SQL的强事务保证和关系模型，重点放在数据库的⾼可⽤性和可扩展性,⾼性能

noSQL 的主要优势：

* ⾼可⽤性和可扩展性，⾃动分区，轻松扩展
* 不保证强⼀致性，性能⼤幅提升
* 没有关系模型的限制，极其灵活

noSQL 不保证强⼀致性，对于普通应⽤没问题，但还是有不少像⾦融⼀样的企业级应⽤有强⼀致性的需求。

⽽且 noSQL 不⽀持 SQL 语句，兼容性是个⼤问题，不同的 noSQL 数据库都有⾃⼰的 api 操作数据，⽐较复杂。

* 1. **newSQL** 特性

newSQL 提供了与 noSQL 相同的可扩展性，⽽且仍基于关系模型，还保留了极其成熟的

SQL 作为查询语⾔，保证了ACID事务特性。

简单来讲，newSQL 就是在传统关系型数据库上集成了 noSQL 强⼤的可扩展性。

传统的SQL架构设计基因中是没有分布式的，⽽ newSQL ⽣于云时代，天⽣就是分布式架构。

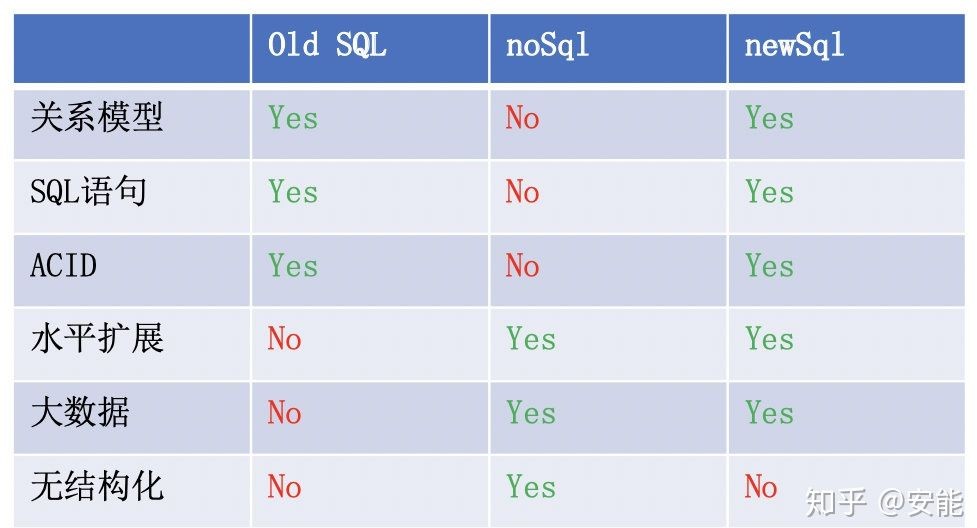
noSQL 的主要特性：

SQL ⽀持，⽀持复杂查询和⼤数据分析(OLAP)。

⽀持 ACID 事务，⽀持隔离级别。

弹性伸缩，扩容缩容对于业务层完全透明。

⾼可⽤，⾃动容灾。



1. **CAP**定理

分布式数据库中**CAP**原理**CAP+BASE**

1.传统ACID

A (Atomicity) 原⼦性

C (Consistency) ⼀致性

I (Isolation) 独⽴性

D (Durability) 持久性

2.CAP

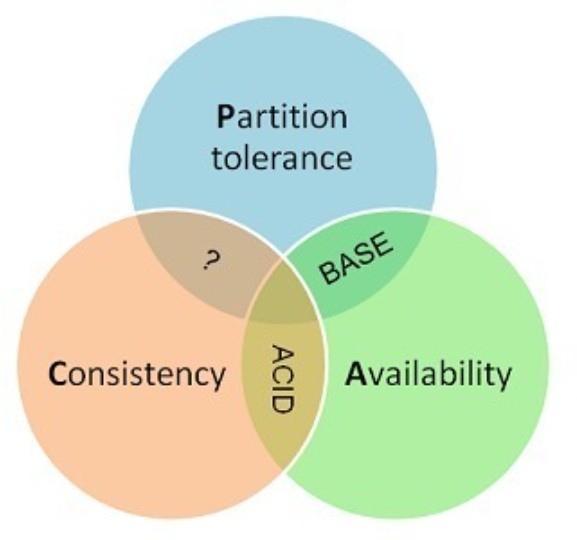
C:Consistency（强⼀致性）：all nodes see the same data at the same time

A:Availability（可⽤性）:Reads and writes always succeed，和分布式数据冗余，负载均衡等有着很⼤的关联

P:Partition tolerance（分区容错性）:the system continues to operate despite arbitrary messag e loss or failure of part of the system,分布式系统在遇到某节点或⽹络分区故障的时候，仍然能够对外提供满⾜⼀致性或可⽤性的服务。

2000年7⽉，加州⼤学伯克利分校的Eric Brewer教授在ACM PODC会议上提出CAP猜想。2年后，⿇省理⼯学院的Seth Gilbert和Nancy Lynch从理论上证明了CAP。之后，CAP理论正式成为分布式计算领域的公认定理。

CAP的3进2



⼀个分布式系统最多只能同时满⾜⼀致性（Consistency）、可⽤性（Availability）和分区容错性（Partition to lerance）这三项中的两项。

==========================================================================================

=============================

C:强⼀致性 A：⾼可⽤性 P：分布式容忍性 CA 传统Oracle数据库

AP ⼤多数⽹站架构的选择

CP Redis、Mongodb

注意：分布式架构的时候必须做出取舍。

⼀致性和可⽤性之间取⼀个平衡。多余⼤多数web应⽤，其实并不需要强⼀致性。因此牺牲C换取P，这是⽬前分布式数据库产品的⽅向

==========================================================================================

=============================

⼀致性与可⽤性的决择

对于web2.0⽹站来说，关系数据库的很多主要特性却往往⽆⽤武之地

数据库事务⼀致性需求

很多web实时系统并不要求严格的数据库事务，对读⼀致性的要求很低， 有些场合对写⼀致性要求并不⾼。允许实现最终⼀致性。

数据库的写实时性和读实时性需求

对关系数据库来说，插⼊⼀条数据之后⽴刻查询，是肯定可以读出来这条数据的，但是对于很多web应⽤来说，并不要求这么⾼的实时性，⽐⽅说发⼀条消息之 后，过⼏秒乃⾄⼗⼏秒之后，我的订阅者才看到这条动态是完全可以接受的。 对复杂的SQL查询，特别是多表关联查询的需求

任何⼤数据量的web系统，都⾮常忌讳多个⼤表的关联查询，以及复杂的数据分析类型的报表查询，特别是SNS类型的

⽹站，从需求以及产品设计⾓ 度，就避免了这种情况的产⽣。往往更多的只是单表的主键查询，以及单表的简单条件分⻚

查询，SQL的功能被极⼤的弱化了。

CAP理论的核⼼是：⼀个分布式系统不可能同时很好的满⾜⼀致性，可⽤性和分区容错性这三个需求，

最多只能同时较好的满⾜两个。

因此，根据 CAP 原理将 NoSQL 数据库分成了满⾜ CA 原则、满⾜ CP 原则和满⾜ AP 原则三 ⼤类：

CA - 单点集群，满⾜⼀致性，可⽤性的系统，通常在可扩展性上不太强⼤。

CP - 满⾜⼀致性，分区容忍必的系统，通常性能不是特别⾼。

AP - 满⾜可⽤性，分区容忍性的系统，通常可能对⼀致性要求低⼀些。

举例

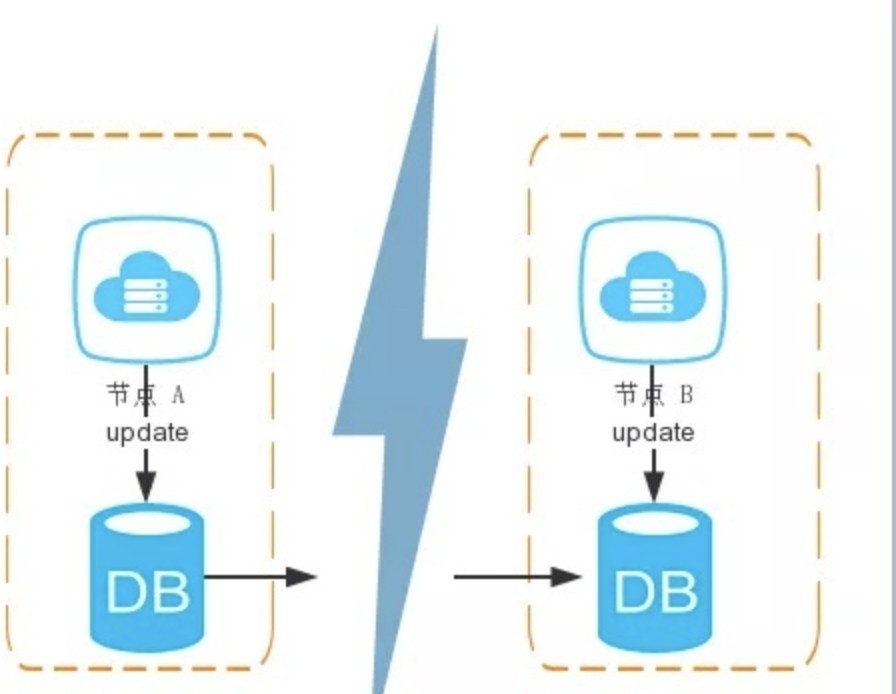
整个系统由两个节点配合组成，之间通过⽹络通信，当节点 A 进⾏更新数据库操作的时候，需要同时更新节点 B 的数据库（这是⼀个原⼦的操作）。

上⾯这个系统怎么满⾜ CAP 呢？

C：当节点A更新的时候，节点B也要更新， A：必须保证两个节点都是可⽤的，

P：当节点 A,B 出现了⽹络分区，必须保证对外可⽤。

可⻅，根本完成不了，只要出现了⽹络分区，A 就⽆法满⾜，因为节点 A 根本连接不上节点 B。如果强⾏满⾜ C 原⼦性，就必须停⽌服务运⾏，从⽽放弃可⽤性 C。



BASE

BASE就是为了解决关系数据库强⼀致性引起的问题⽽引起的可⽤性降低⽽提出的解决⽅案。 BASE其实是下⾯三个术语的缩写：

基本可⽤（Basically Available）

软状态（Soft state）

最终⼀致（Eventually consistent）

它的思想是通过让系统放松对某⼀时刻数据⼀致性的要求来换取系统整体伸缩性和性能上改观。为什么这么说呢，缘由就在于⼤型系统往往由于地域分布和极⾼性能的要求，不可能采⽤分布式事务来完成这些指标，要想获得这些指标，我们必须采

⽤另外⼀种⽅式来完成，这⾥BASE就是解决这个问题的办法

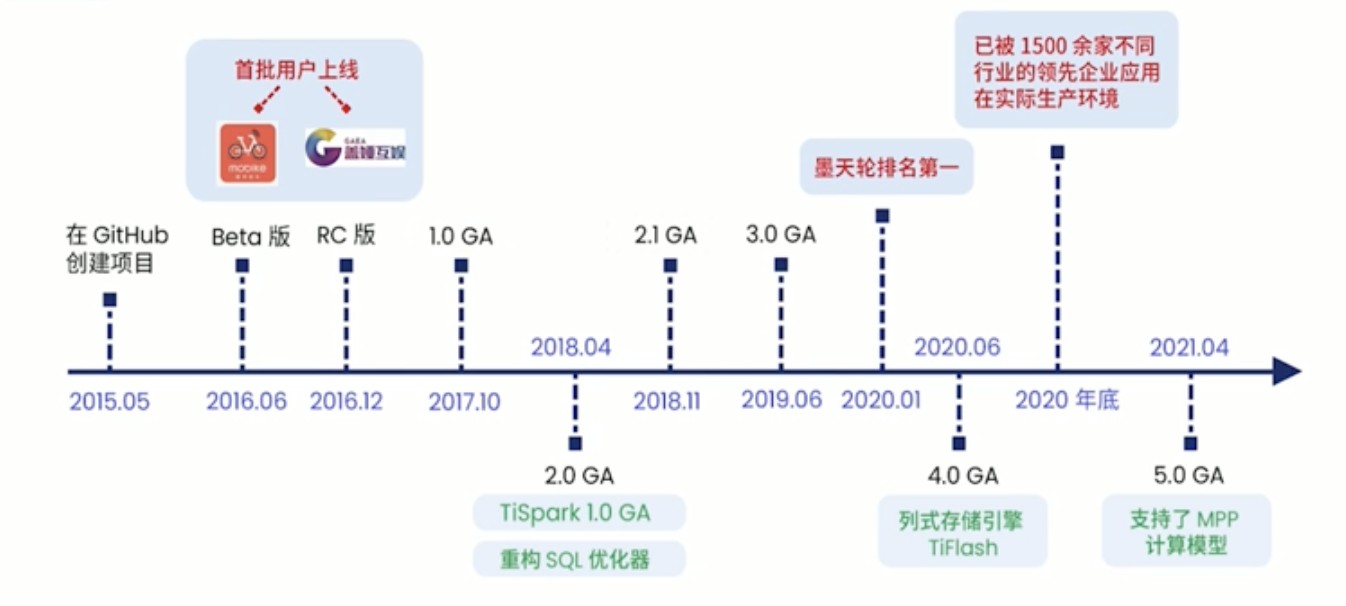
1. **TiDB**简介

分布式系统简介

1. 集中式系统⽆法承载所需要的计算量
2. 分布式系统:不同组件,⼤量机器协调⼯作,通过冗余和分段进⾏功能组合
   * GFS:分布式⽂件存储
   * Google Bigtable:key存储
   * MapReduce：分析和计算
3. 主要挑战-CAP
   * 最⼤程度实现分治
   * 全局⼀致性
   * 故障与容错
   * ⽹络分区问题

4.NewSQL:关系模型(强事务)+分布式系统+SQL

TiDB版本迭代



* 1. **TiDB**整体概述

特点

1. 扩展性:写⼊能⼒与容量
2. 强⼀致+⾼可⽤:保证⽹络延时和数据容灾 RPO:数据点位恢复丢失数据量

RTO:数据业务恢复时间

1. 标准SQL+⽀持ACID事务
2. 云原⽣:弹性设计
3. HTAP:TP+AP,⾏列混合

数据基础技术

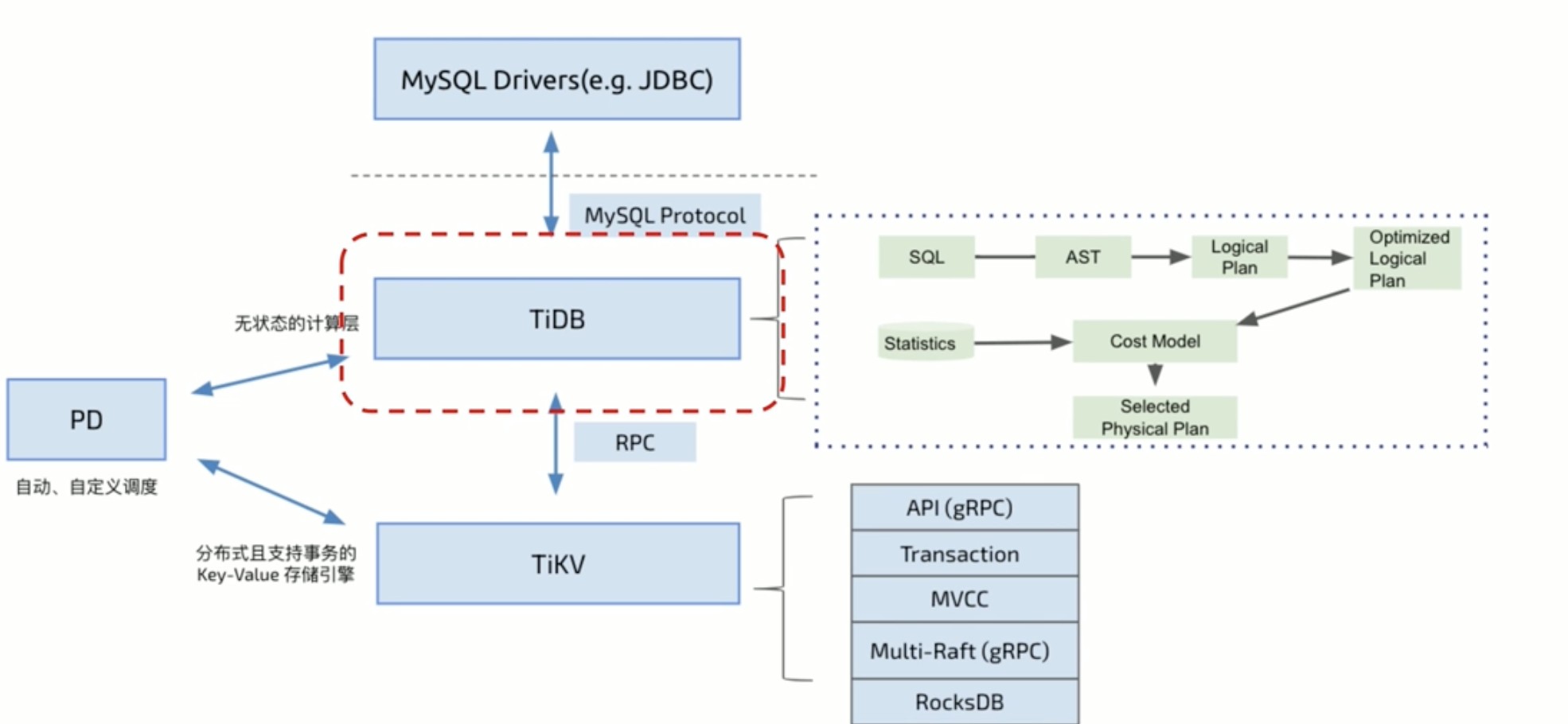
1. 数据模型:Kv
2. 数据存储与检索:LSM+Tree
3. 数据格式:结构化数据
4. 存储引擎:RockDB 5.复制协议:raft

6.分布式事务模型:Percolator 7.数据架构:Share-Disk

1. 优化器算法:执⾏计划⽣产
2. 执⾏引擎:向量化
3. 计算引擎:标准SQL引擎
4. 分布式技术:存算分离,解耦保证资源利⽤最⼤化和弹性扩缩容

TiDB⾼度分层架构

TiDB:计算 TiKV:存储 PD:调度与管理

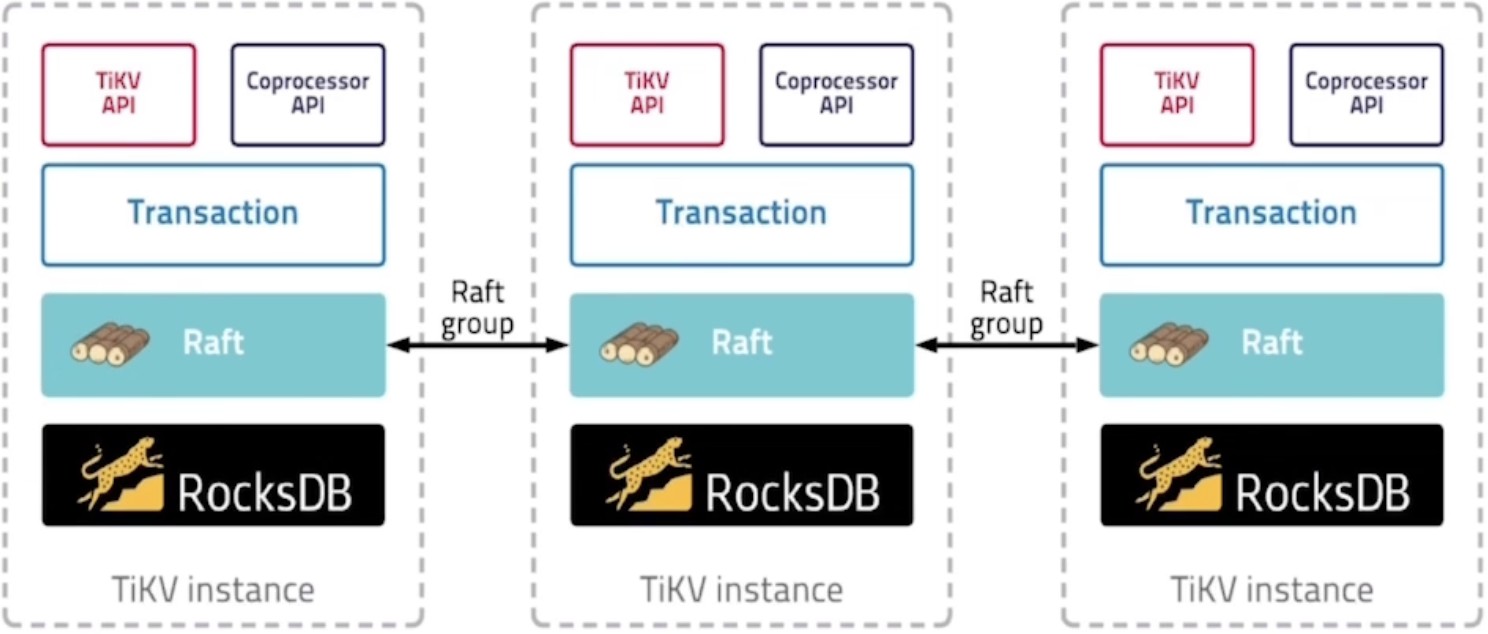


* 1. **TiKV**存储引擎简介

特性需求

1. 弹性扩缩容
2. ⾼可⽤与多副本⼀致性
3. ⾼并发读写
4. 分布式事务
5. 数据准确性

TiKV架构



数据结构

1. 选择

B+tree写两次:WAL+写⼊树,额外的分裂与平衡且写⼊容量上限

kv结构+LSM-Tree:空间换取写⼊延时,顺序写⼊替换随机写⼊和调整代价

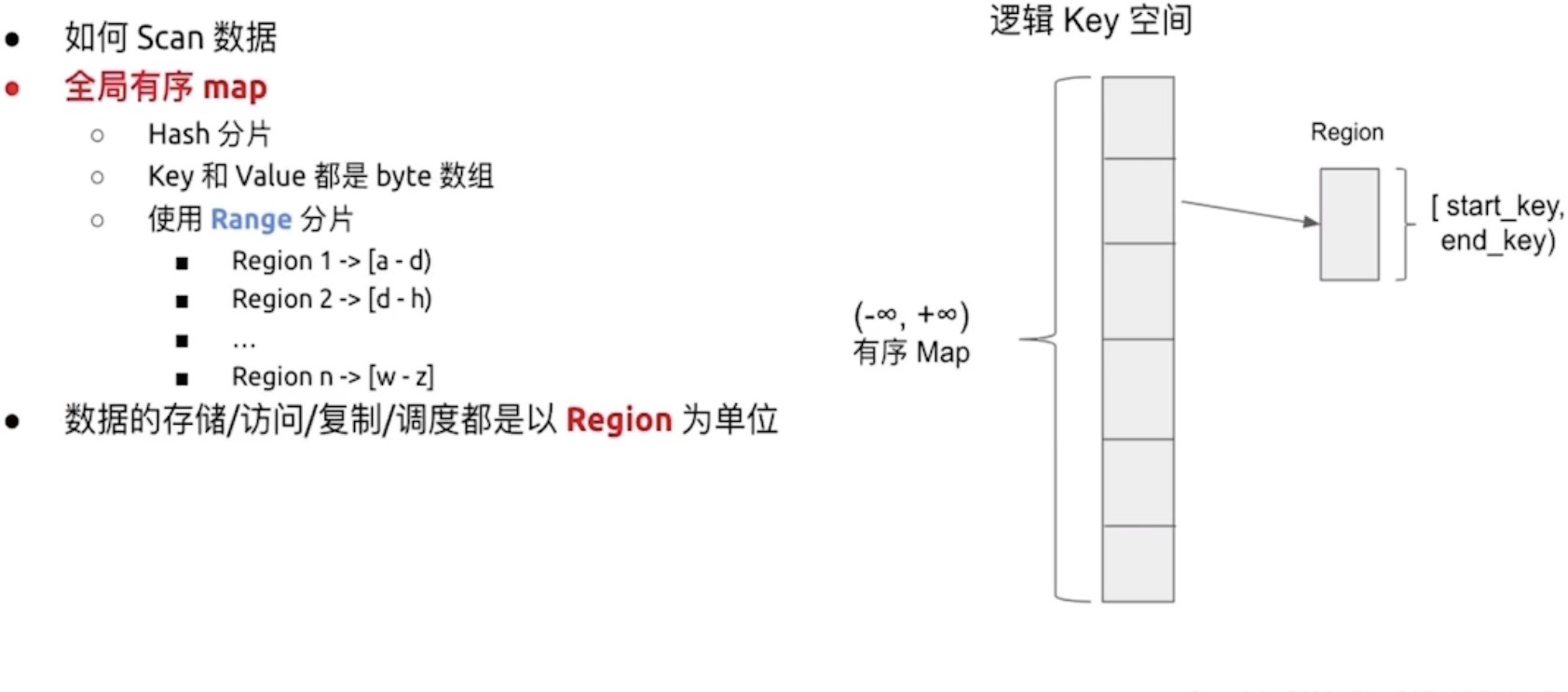
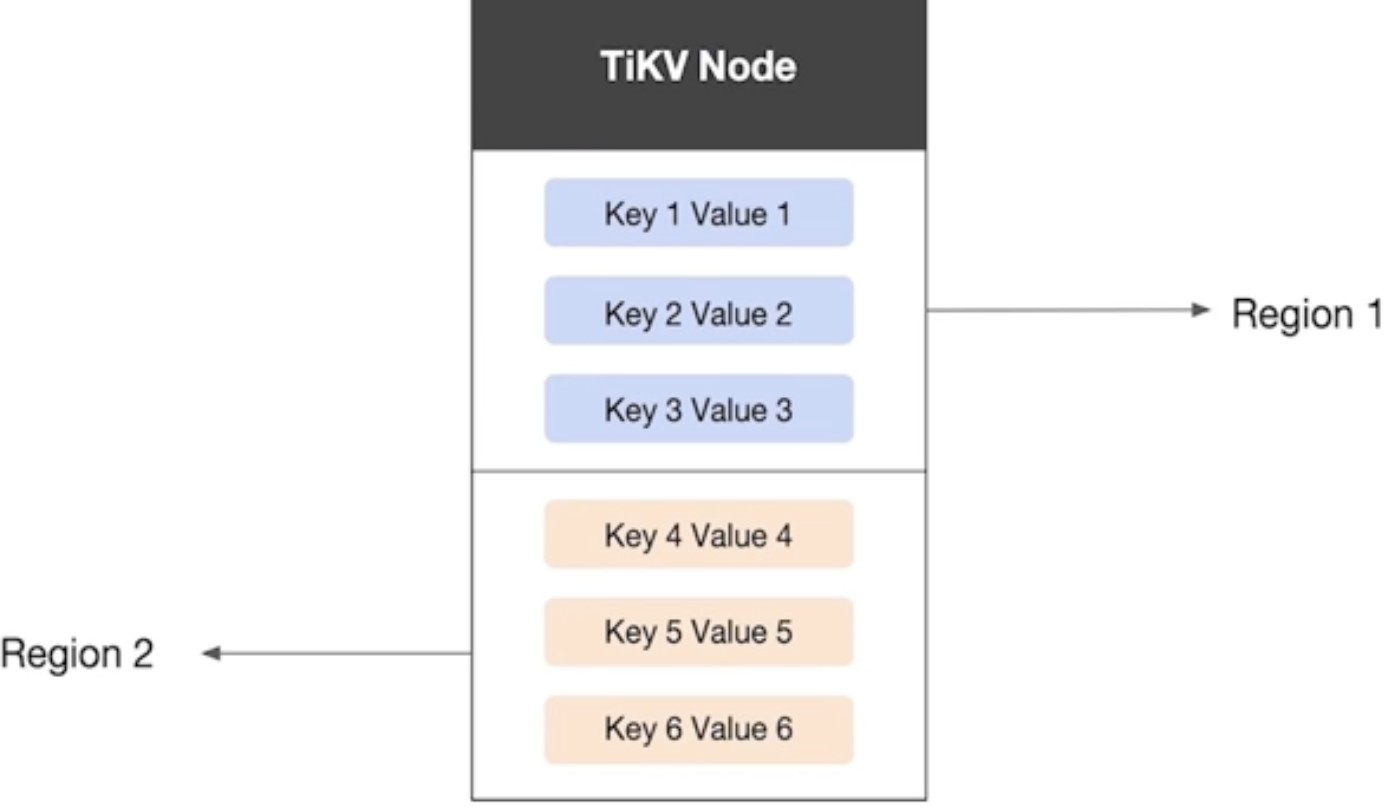
1. RocsDB
   * LSM tree
   * bathch write
   * snapshot

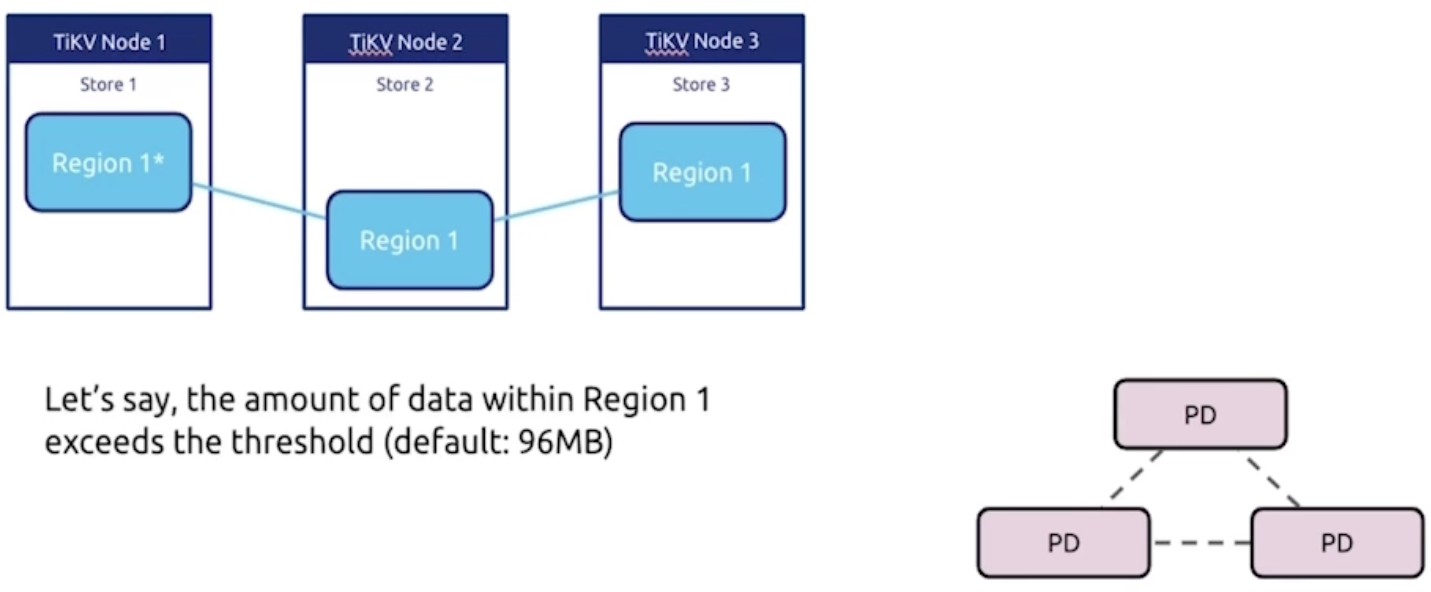
分布式复制协议

Raft:⽐Paxos逻辑更简单,模块和简化,易实现 Raft+rockdb:⾃动分⽚(Range)+强⼀致性

分⽚概念

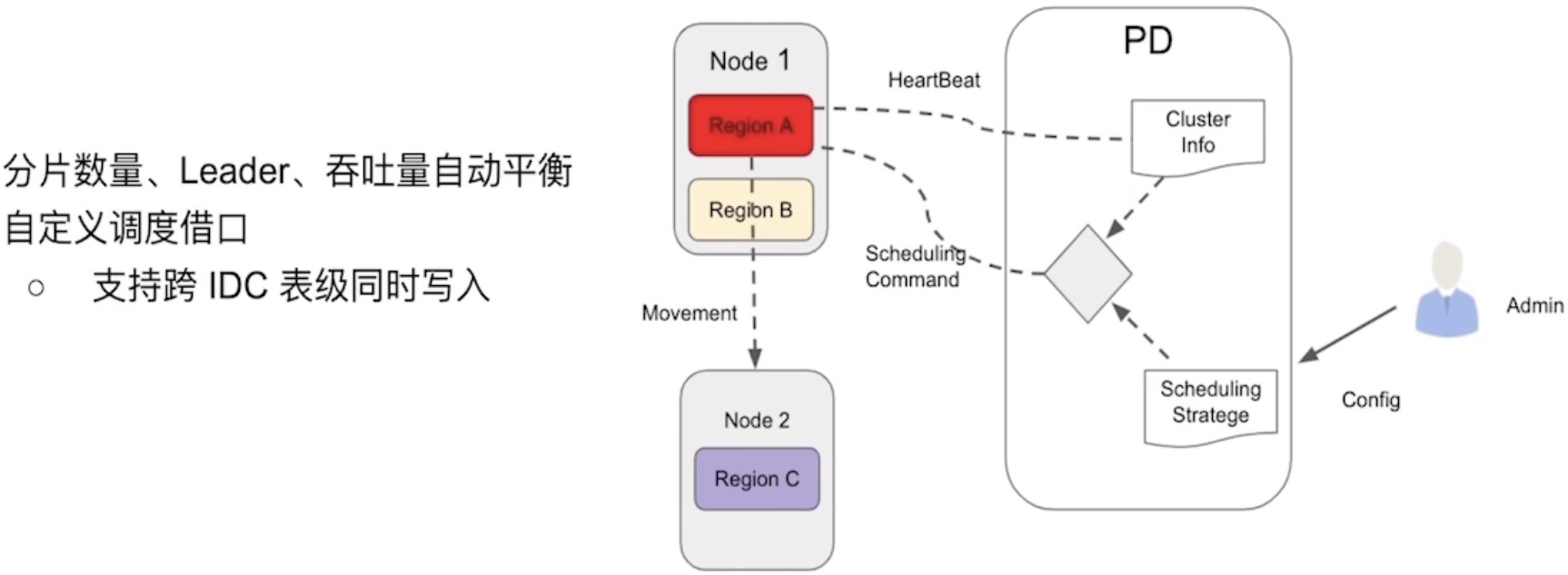
[start1,end1) [start2,end2)





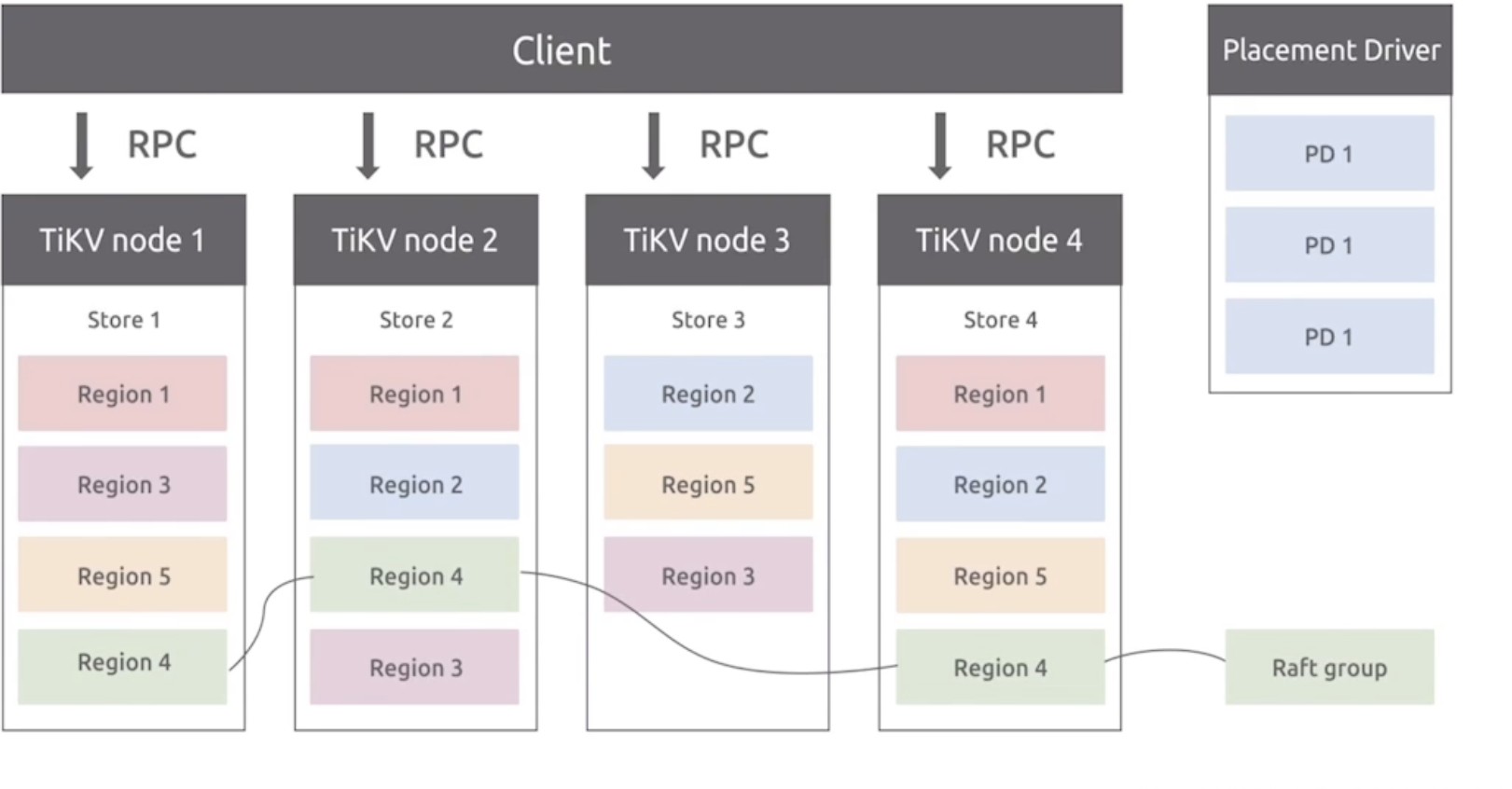
灵活调度

PD进⾏调度,通过⼼跳判断负载



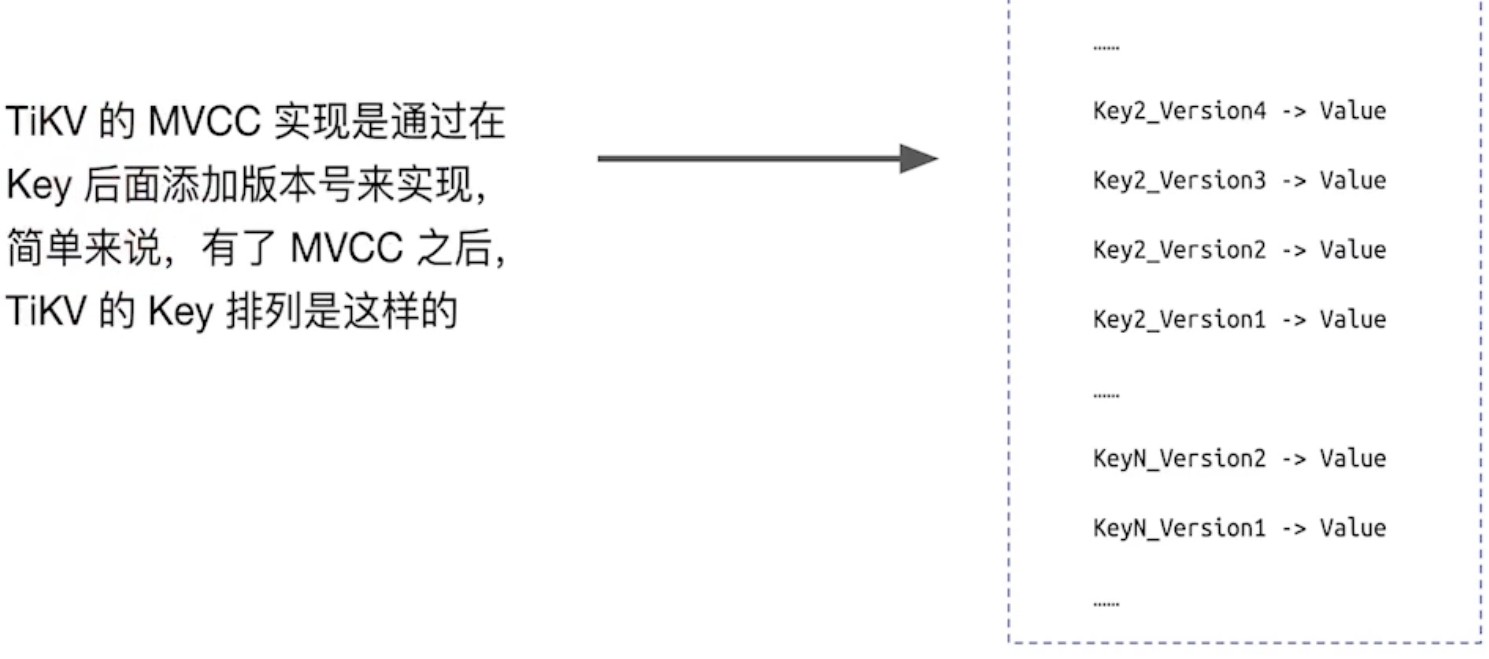
TiKV架构

1.raft复制三副本,leader提供读写 2.pd进⾏均衡调度保证负载均衡

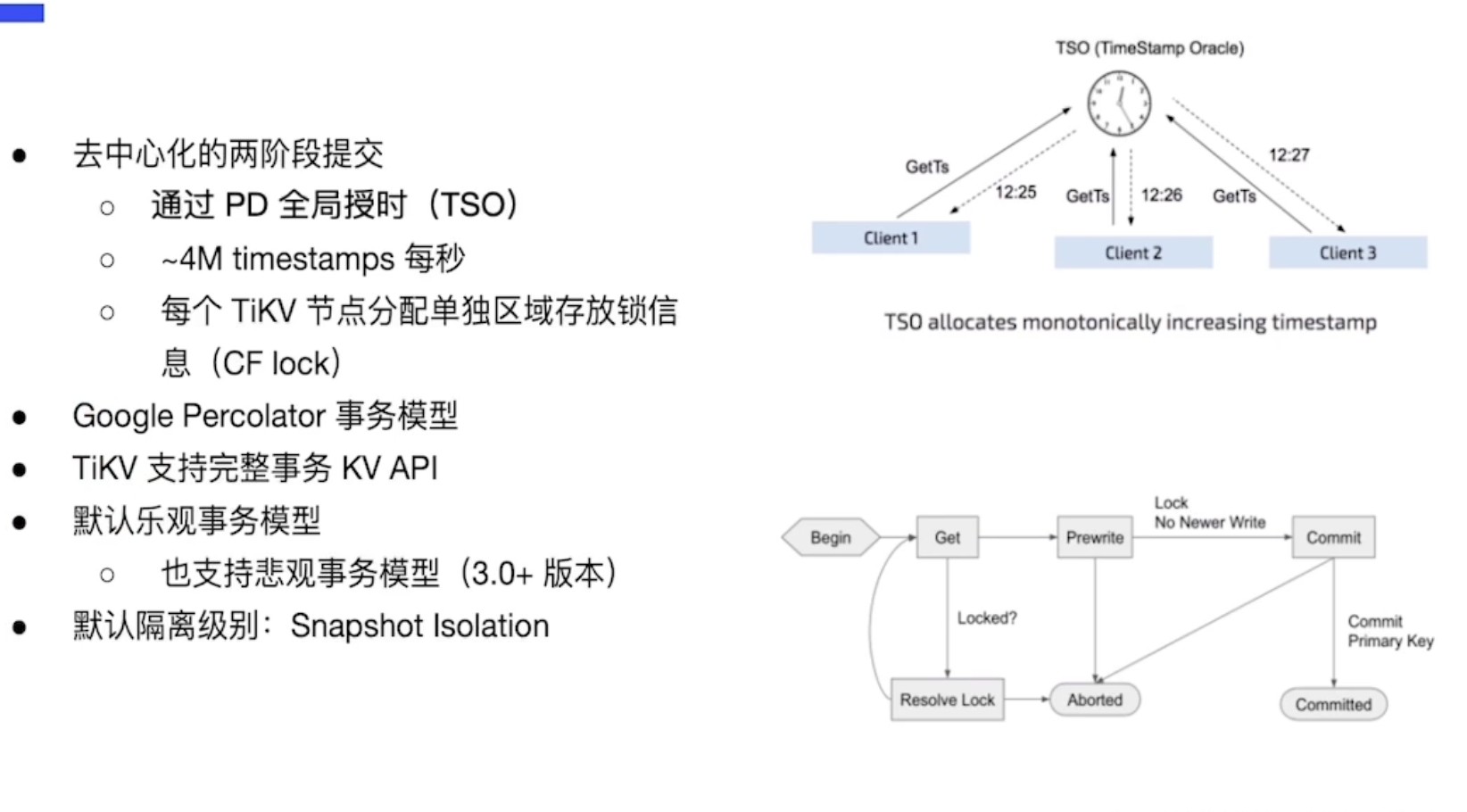


MVCC

SI隔离级别+历史数据恢复

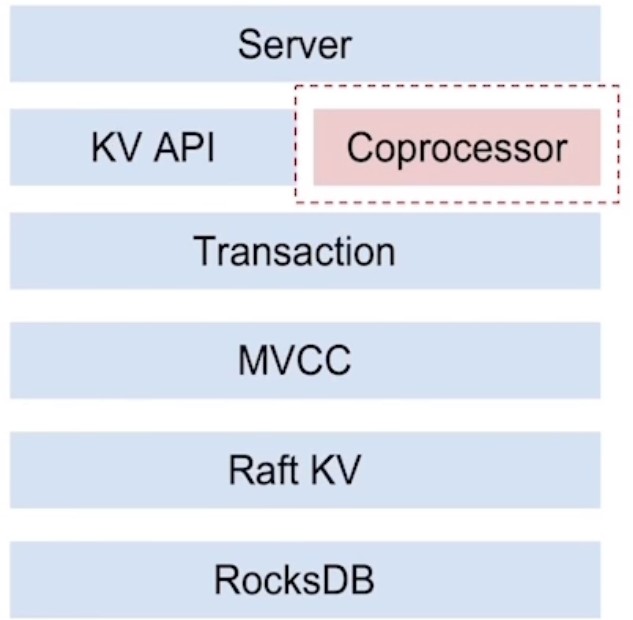


分布式事务



Coprocessor（协作处理器）

- tikv读取数据并计算的模块,每个tikv都有⼀个Coprocessor

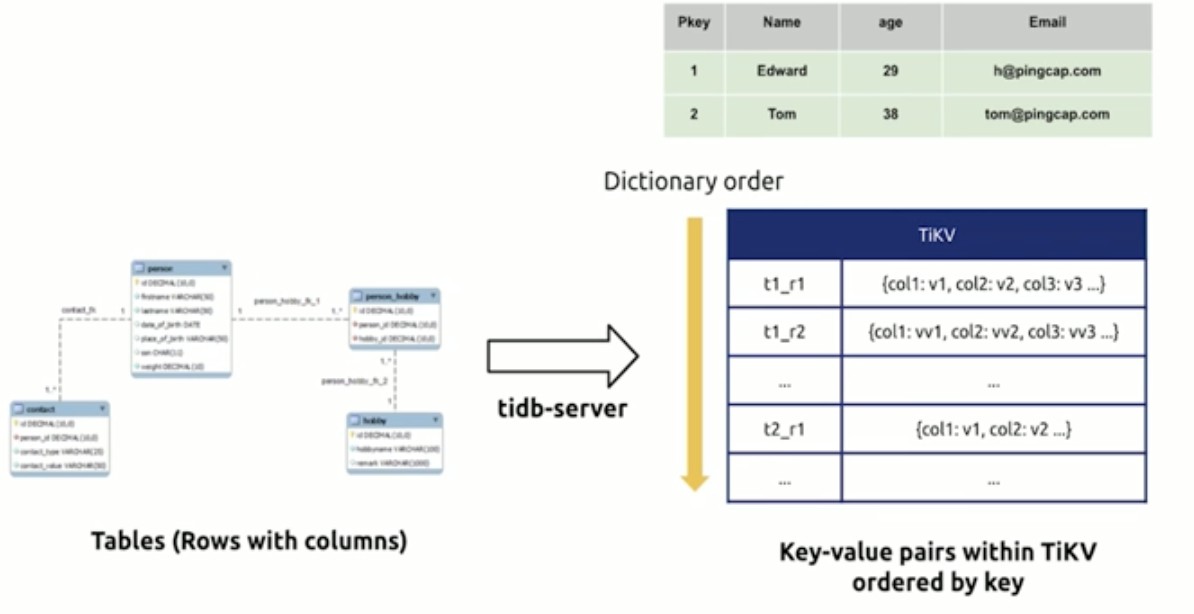


# TiDB-SQL引擎

* + 1. 逻辑表实现

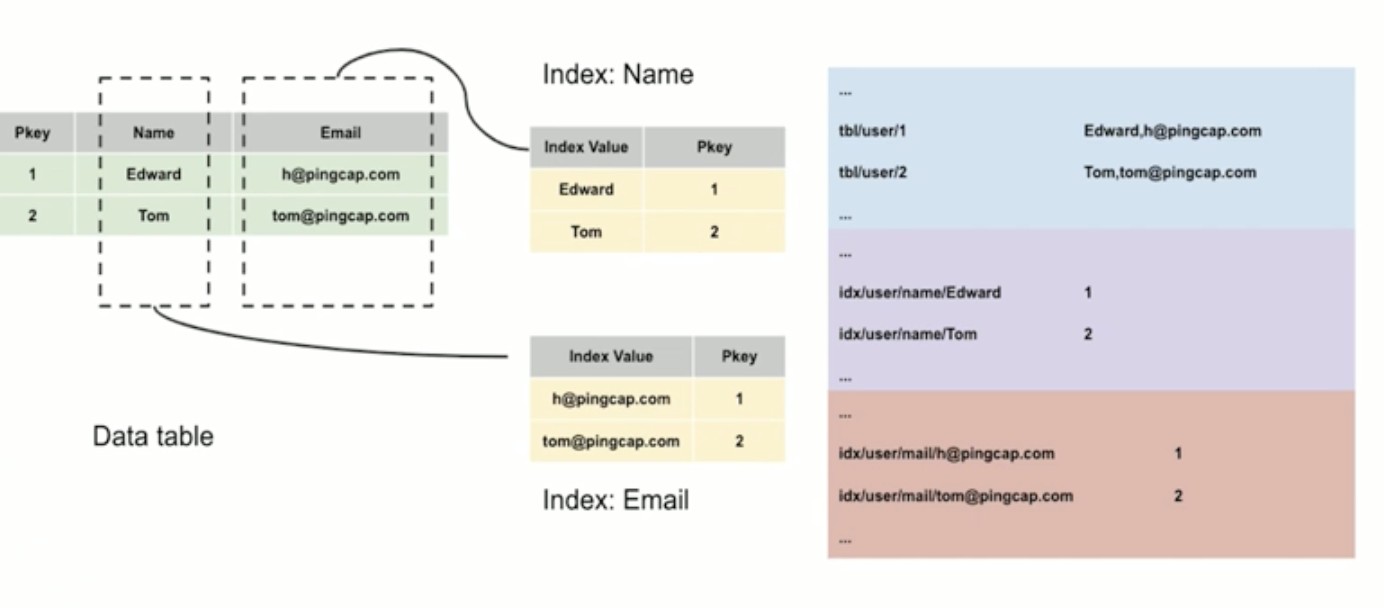
key-value-->table

每个表:tableid\_rowid + value-->映射成table表结构



KV⼆级索引设置

1.key:索引列信息+value:table主键-->原表的key-value



SQL引擎过程

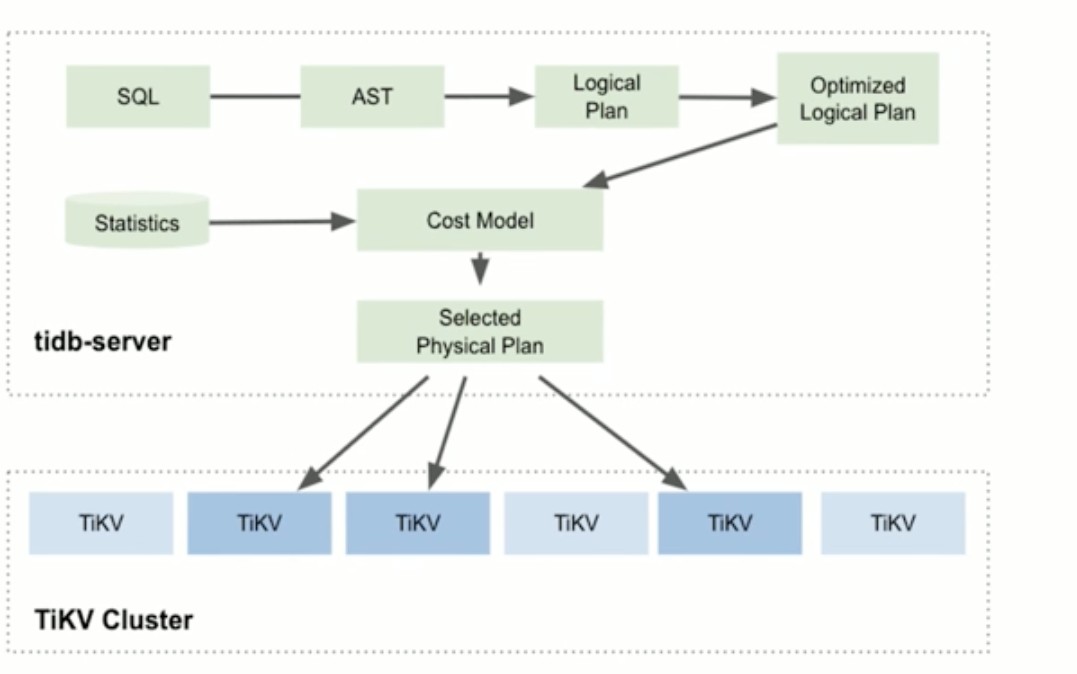
1.sql语法,语义解析 2.ast解析

3.逻辑优化改写

4.物理优化:根据成本优化

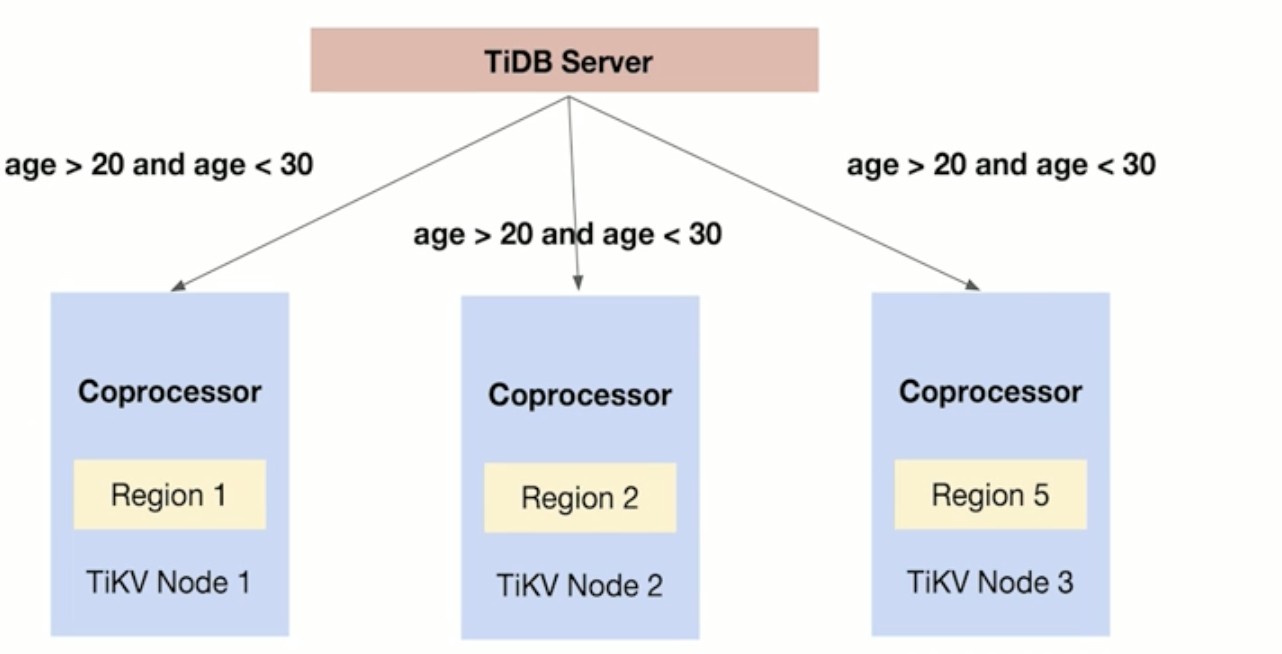
- 根据cup,io等选择最佳执⾏计划

5.分发执⾏器,到tikv执⾏

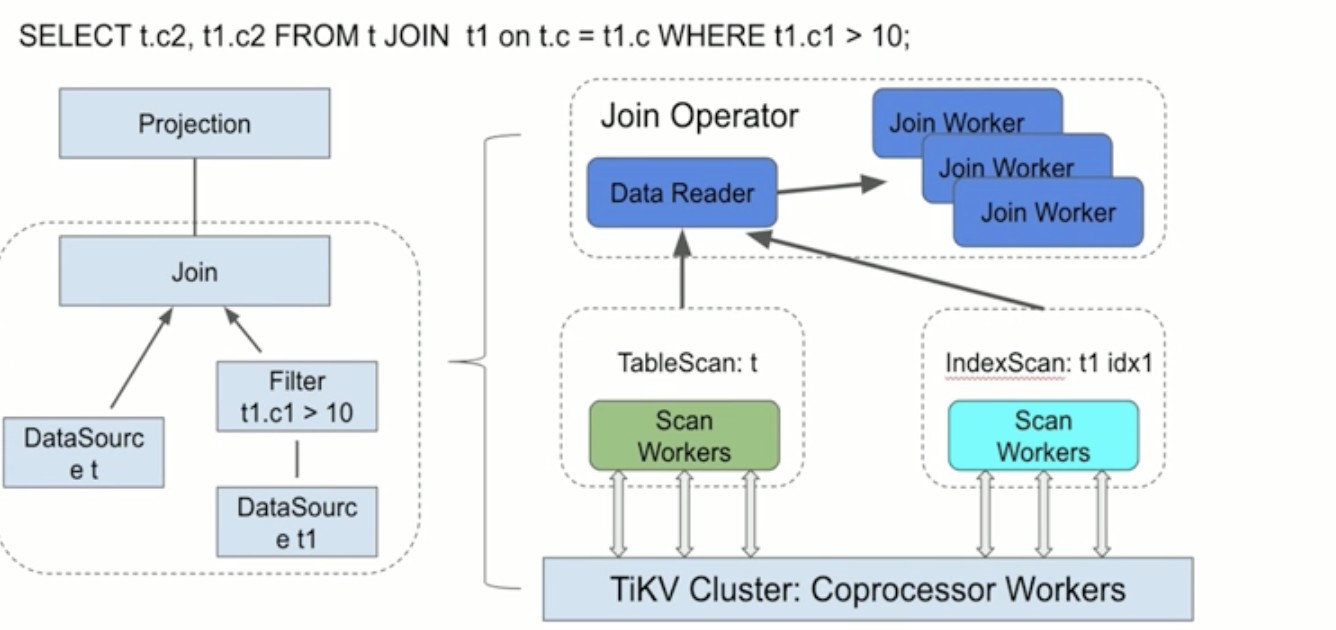


优化策略

* tikv层做好本地预计算
* 上报tidb层进⾏计算聚合

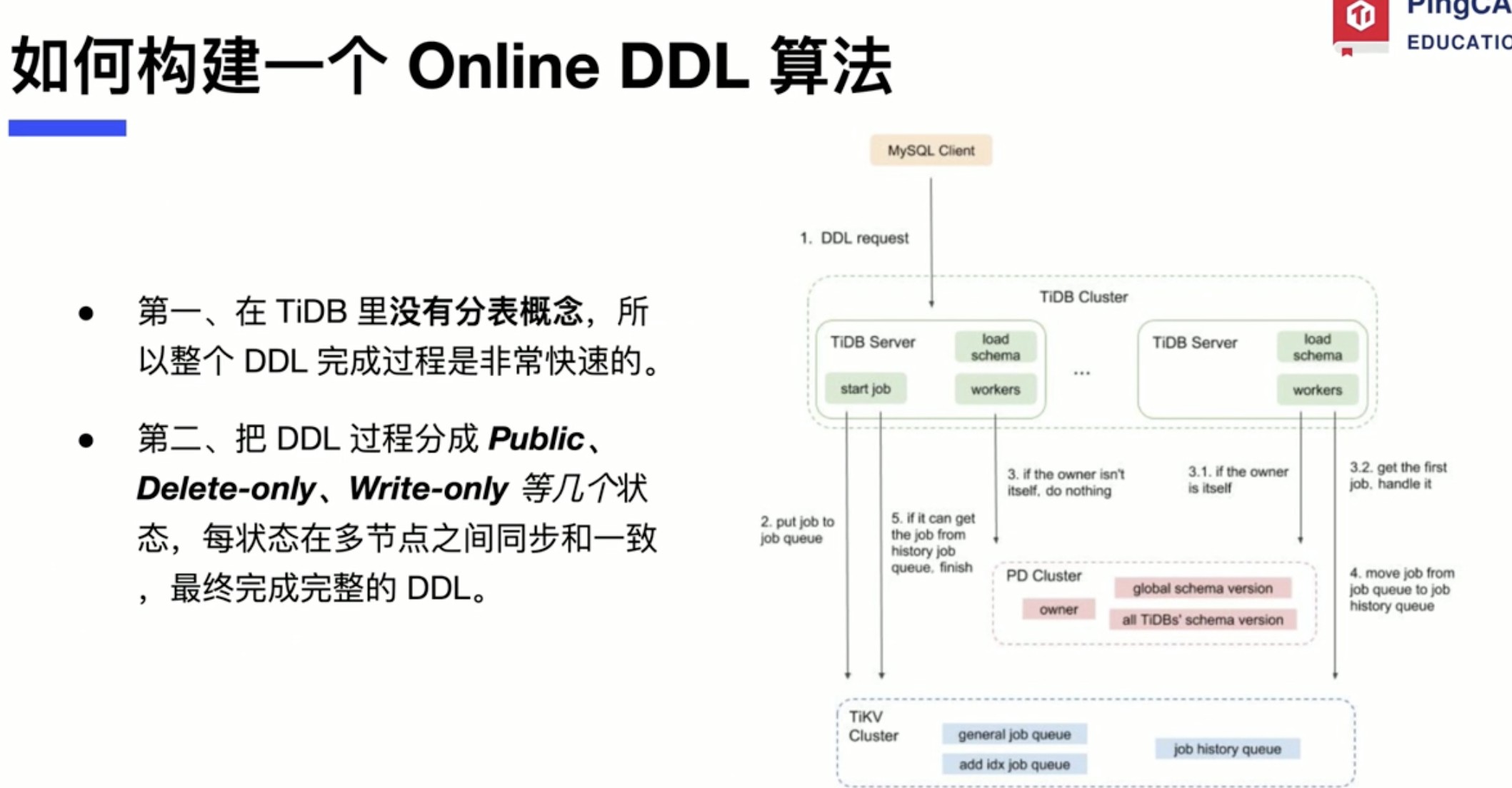


关键算⼦分布式



复杂算⼦尽量推给存储节点(tikv,tiflash),减轻tidb压⼒,分布式计算加快计算效率

online ddl

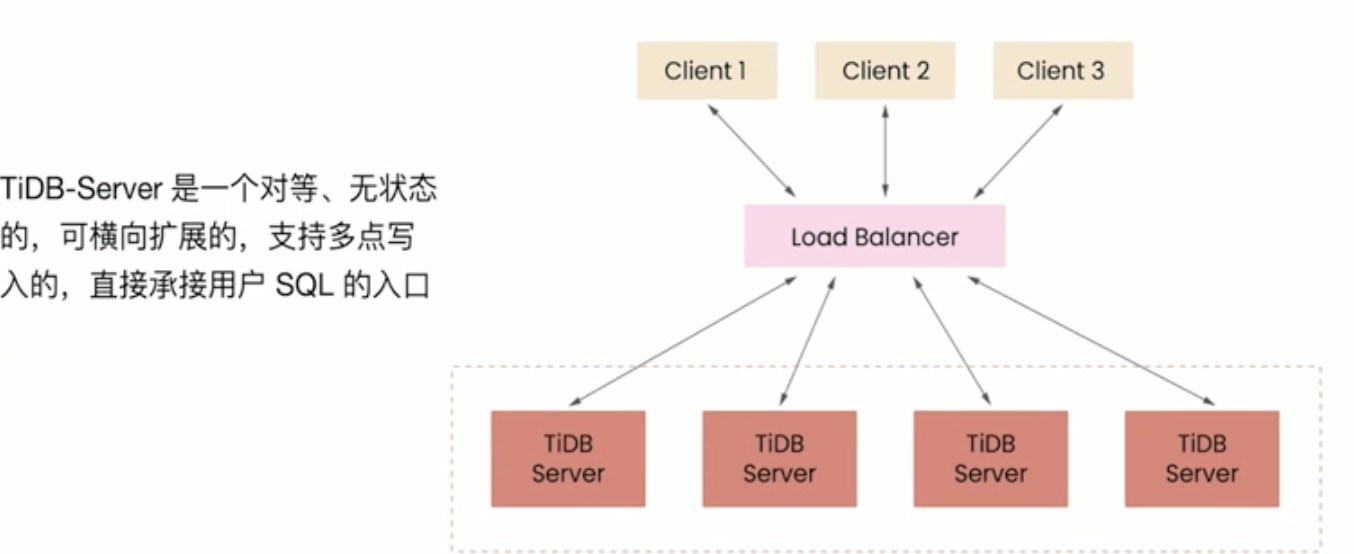


1.对业务⽆重⼤影响,对表结构⽆影响(因为本质是kv)

- 新增字段,默认是0,新数据按照新schema存储,⽼数据在变更和读取才进⾏schema变更,达到schema变更解耦

TiDB-Server负载均衡

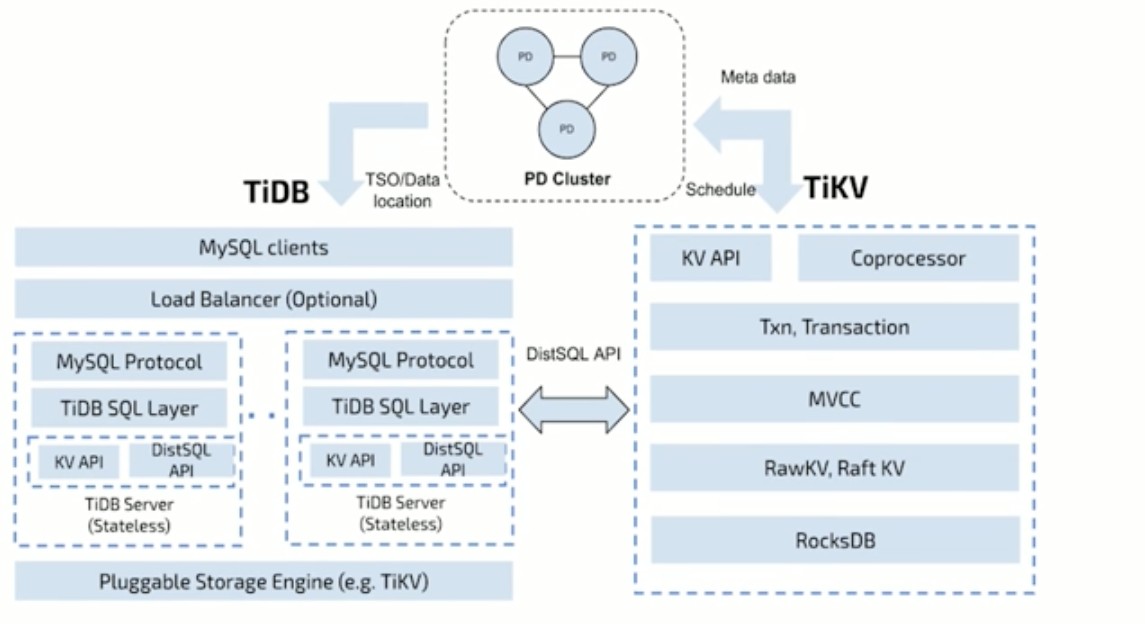
LB进⾏流量转发



tidb-server功能总结



TiDB基础架构

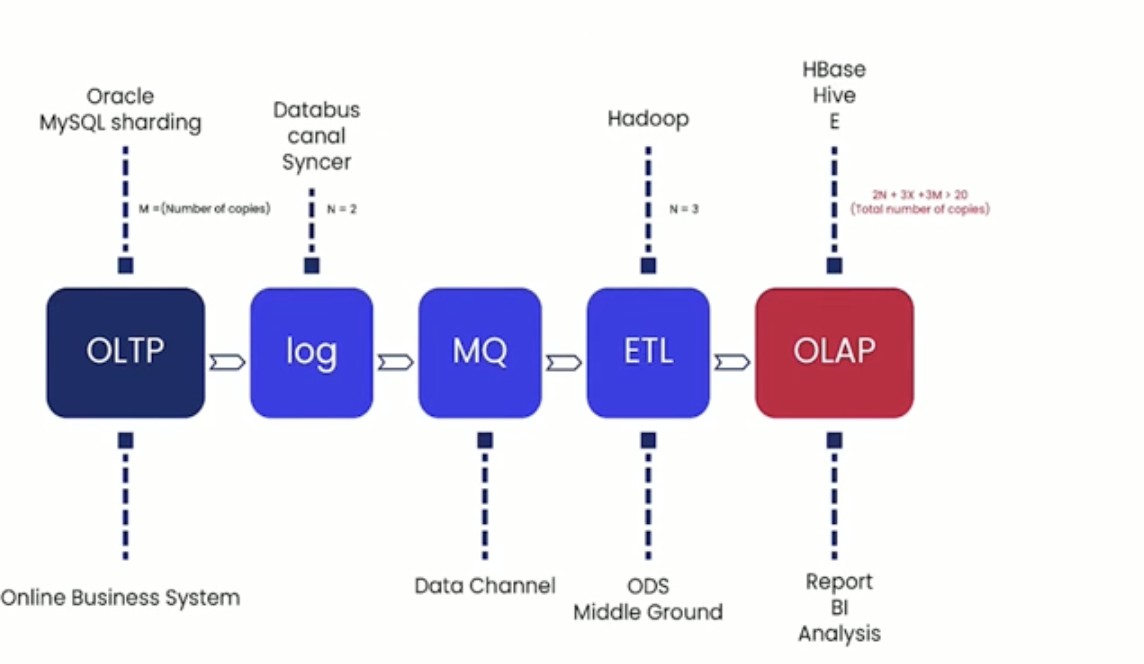


# TiDB-HTAP介绍

正常的ap计算框架

TP：⾼并发,低延时

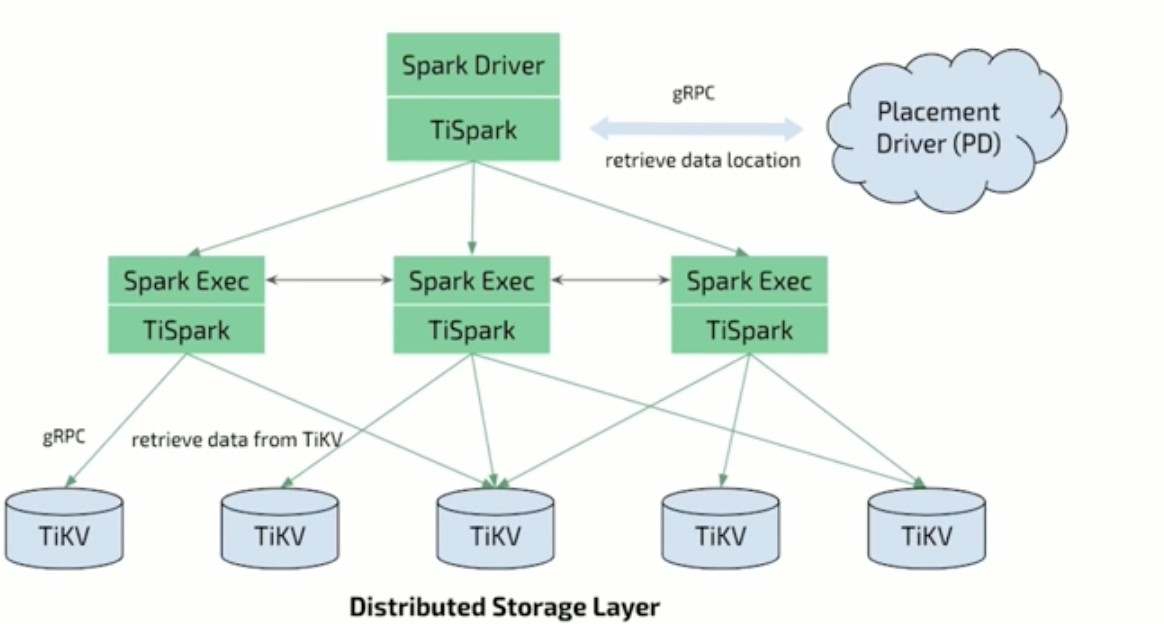
AP：吞吐量,数据切割与分析



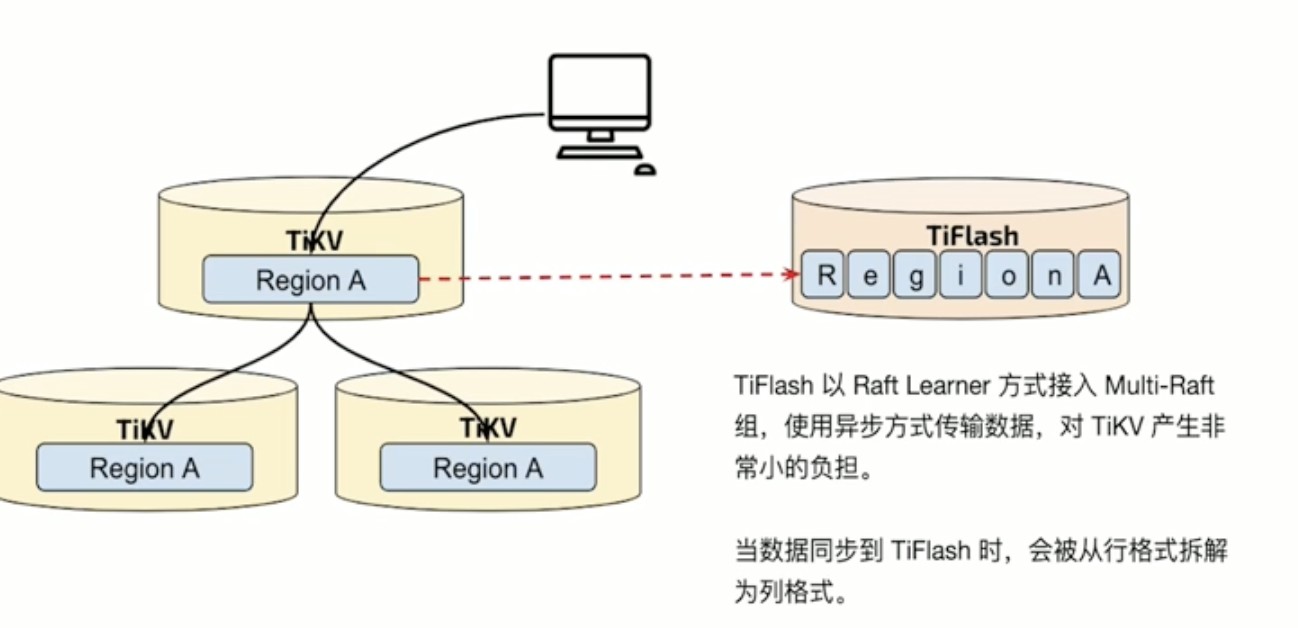
HTAP

* 分布式数据库同事满⾜TP和AP需求,模糊AP和TP使⽤界限
  + 早起TiDB定位解决⼤数据量级的OLTP系统
  + AP需要引⼊新的组件模块
* 引⼊tispark,从tikv中搭建计算引擎
  + 低并发,重量级
* 引⼊tiflash
  + 列存对OLAP友好,选择列存引擎进⾏AP计算
  + 引⼊ck底层存储引擎deltaMerge优化tiflash
  + 将tikv副本通过raft进⾏准实时的tiflash副本更新,tiflash副本仅同步不参与选举

01 NewSQL简介 23

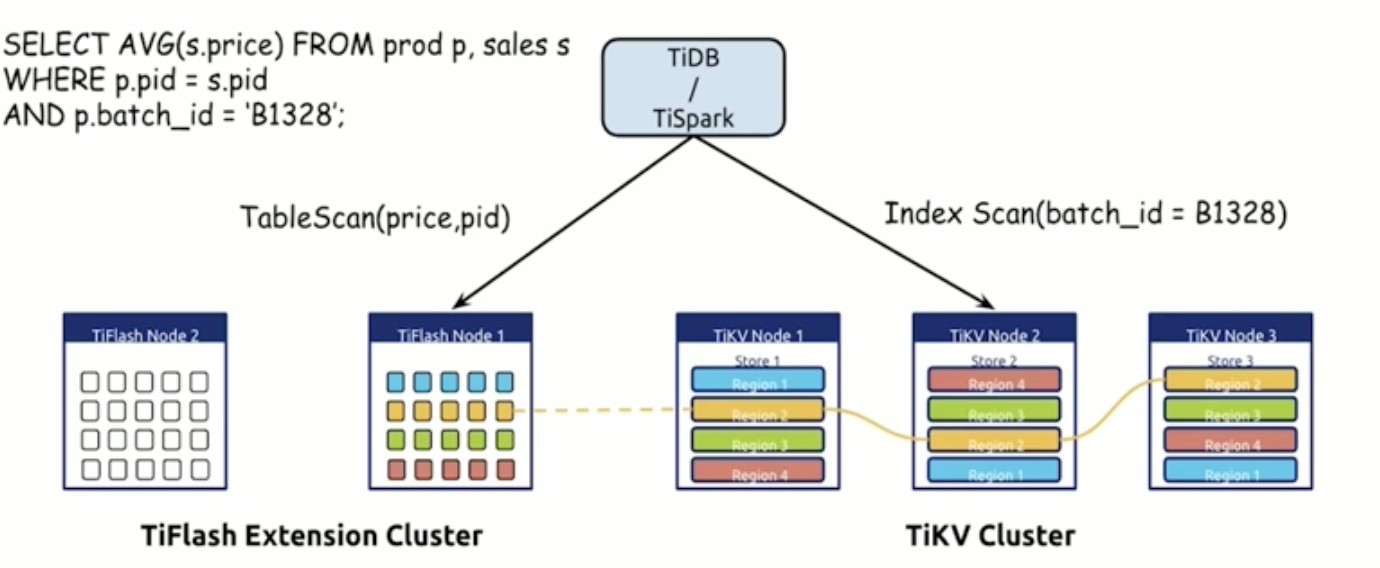


tiflash同步tikv副本进⾏列存转化



Tiflash加⼊后

1. 表关联:tidb进⾏数据join
2. 表过滤:tikv有⼆级索引 batch\_id，tispark寻址 3.聚合:列存进⾏聚合



引⼊tiflash MPP

MPP:并⾏计算,汇聚结果,⽐如join 。

* mpp在tiflash节点将多表关联结果数据保持分布⼀直
* worker负责在多个节点计算上⼀步的分割表关联结果
* 结果返回tidb-server聚合在计算

