TiDB 原理与实战

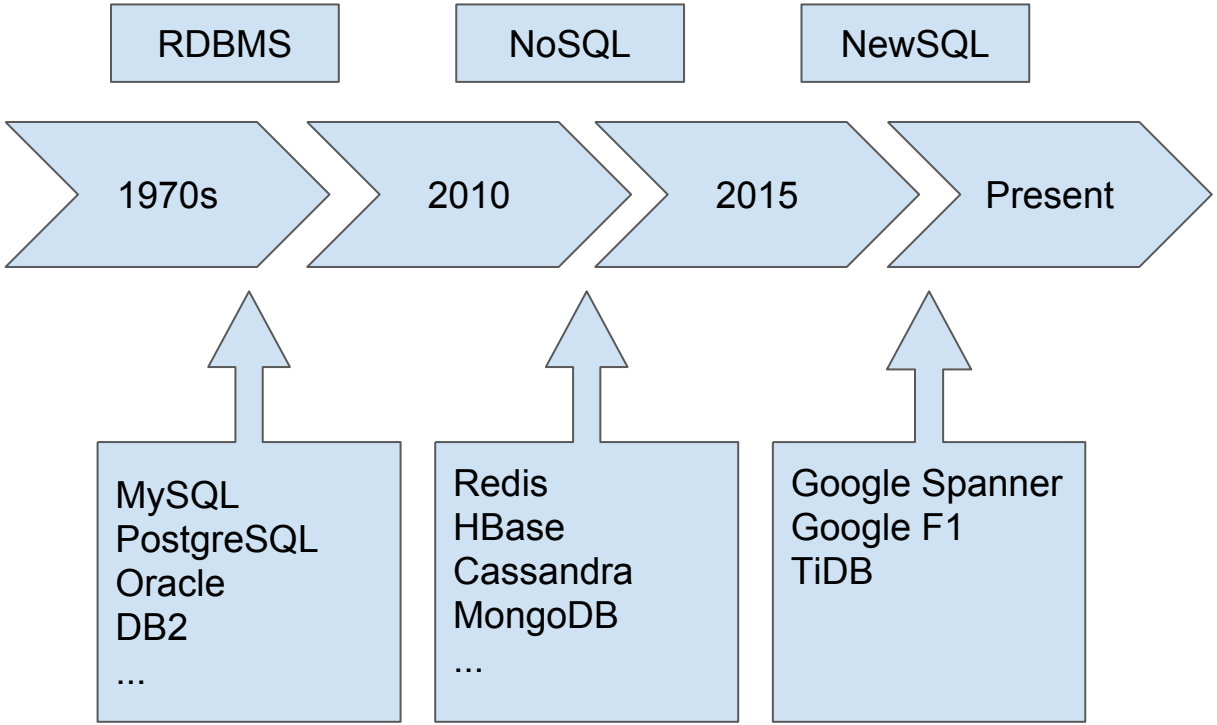
About me

- 程序媛，TiDB committer，Go 语言狂热粉
  - 主要研究方向为分布式系统，坚信分布式系统才是未来
- 目前在 PingCAP 就职
  - 15 年中旬加入 PingCAP
  - 主要参与模块为 TiDB 的 online DDL，SQL 优化器，各种必要的功能改 进以及性能提升
- 之前在京东就职
  - 12 年末入职，学习 go 并且开始做云推送 项目
  - 13 年末开始做存 储方面的工作，云存 储和弹性块存储项目

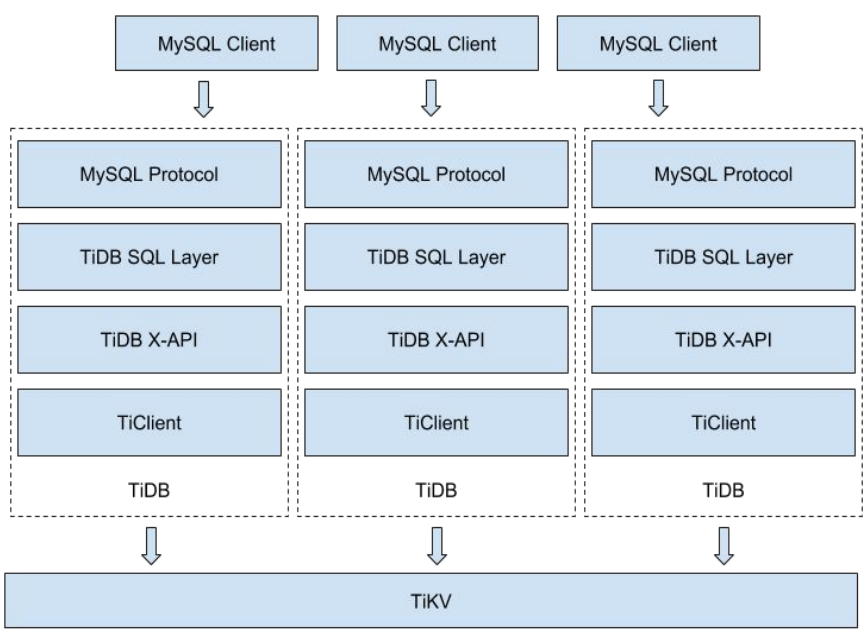
Agenda

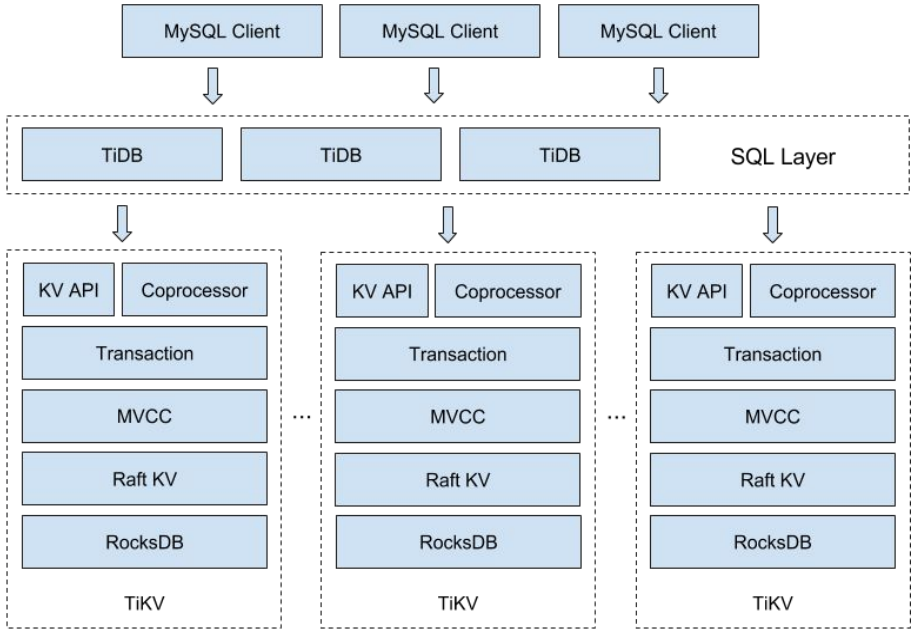
- A brief introduction of NewSQL
- TiDB
- Plan optimization
- Dist SQL
- Online DDL
- TiKV
- Feelings
- Q & A

A brief introduction of NewSQL



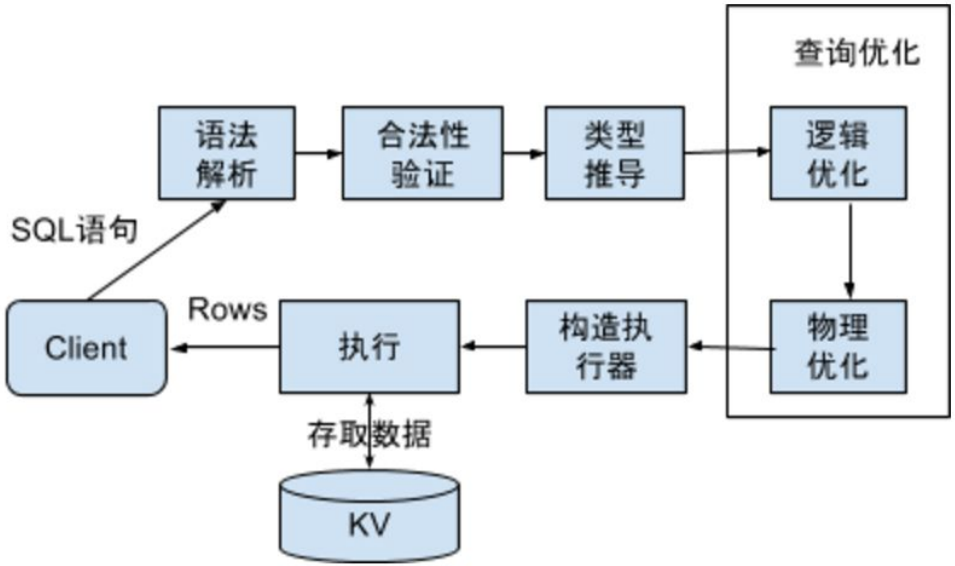
TiDB and TiKV





TiDB

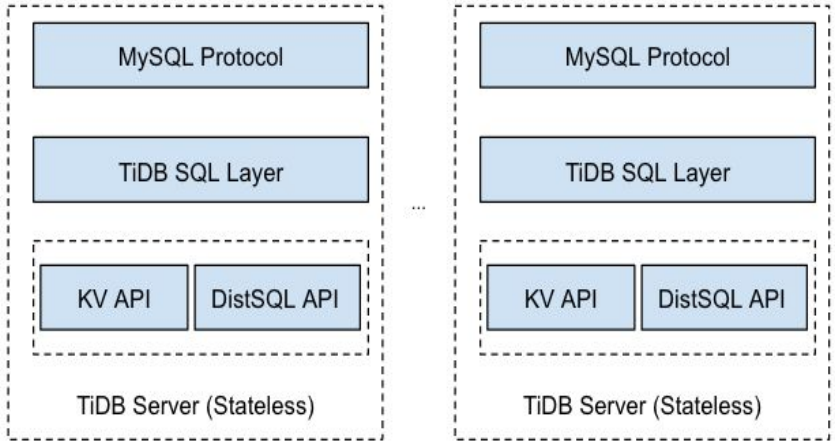
执行流程:


查询优化
语法解析
合法性验证
类型推导
逻辑优化
SQL语句
Client
Rows
执行
构造执行器
物理优化
存取数据
KV

TiDB

支持 MySQL 协议

- 用户从 MySQL 的相关解决方案迁移 过来时几乎没迁移成本。
- 目前还有少量函数或功能未 实现



Plan optimization

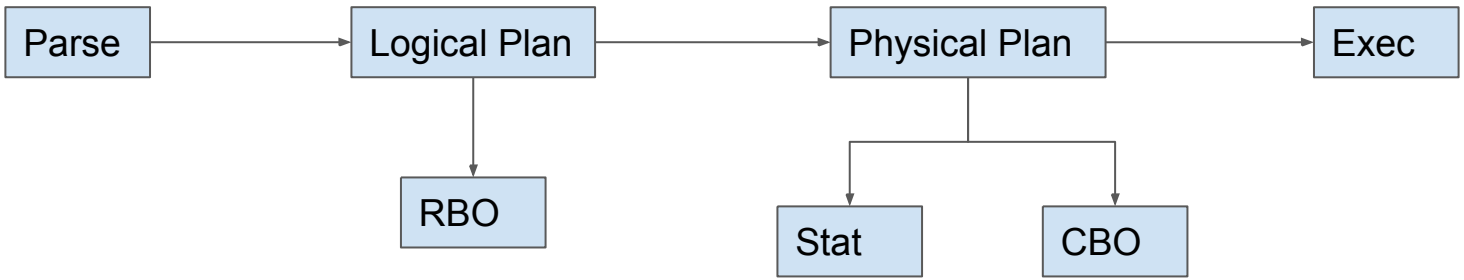
逻辑优化

- 主要依据关系代数的等价交 换做一些逻辑变换

物理优化

- 主要依据数据 读取、表连接方式、表 连接顺序、排序等技 术对查询进行优化。

TP



Plan optimization

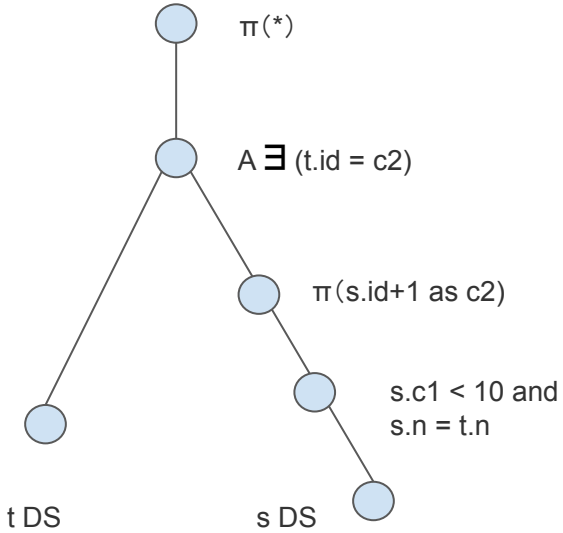
Logical plan

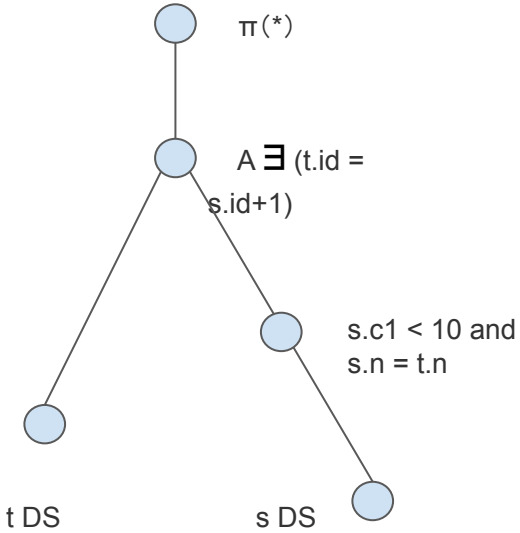
- Prune column
  - select * from s where 0 = (select count(*) from t group by s.id) ->
    select * from s where 0 = (select count(*) from t group by 1)
- Decorrelation
- Predicate push down
- Eliminate aggregation
  - select min(b)from t group by id -> select b from t id is unique index
  - select count(*)from t group by id -> select 1 from t id is unique index
- Aggregation push down

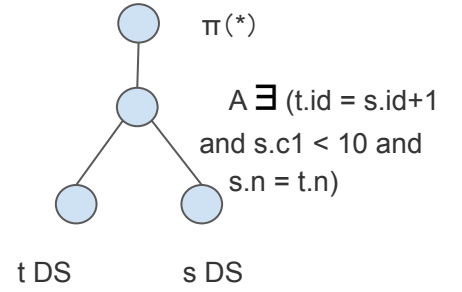
Plan optimization

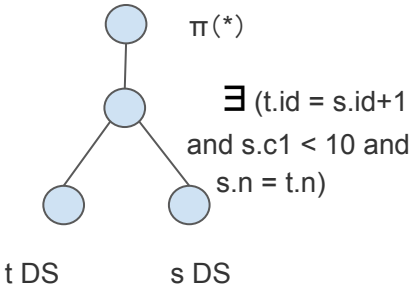
Decorrelation

select * from t where t.id in (select id+1 as c2 from s where s.c1 < 10 and s.n = t.n)





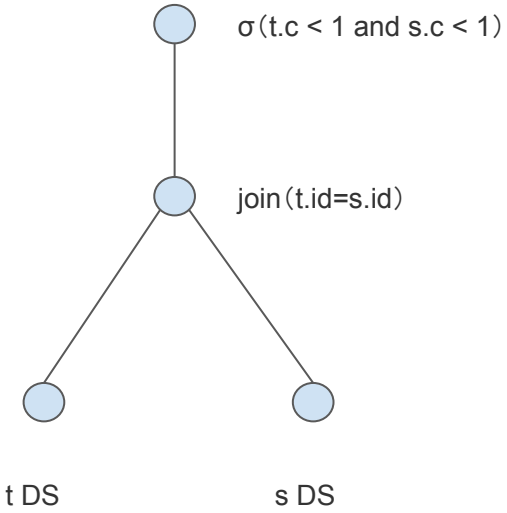


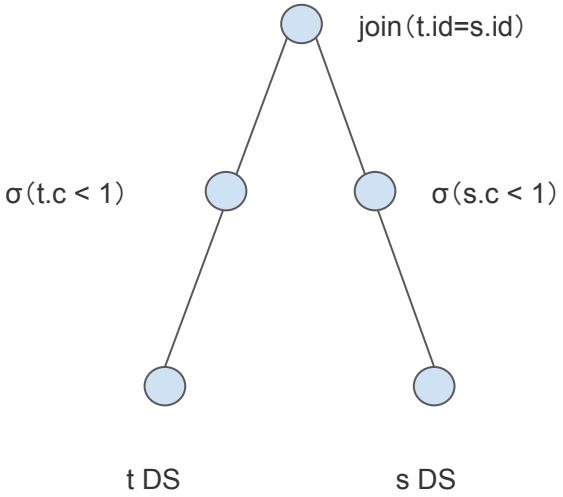


Plan optimization

Predicate push down

select * from t join s on t.id = s.id where t.c < 1 and s.c < 1
如若改成 left out join 则只能下推 σ(t.c < 1)

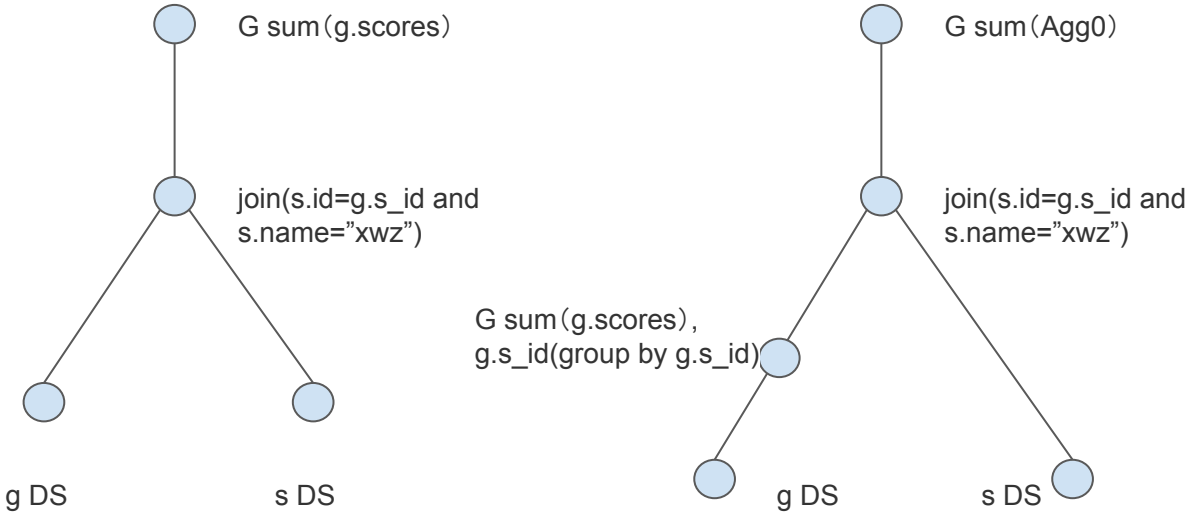




Plan optimization

Aggregation push down

select sum(grade.scores) from stu join grade on stu.id = grade.stu_id and stu.name="xwz"

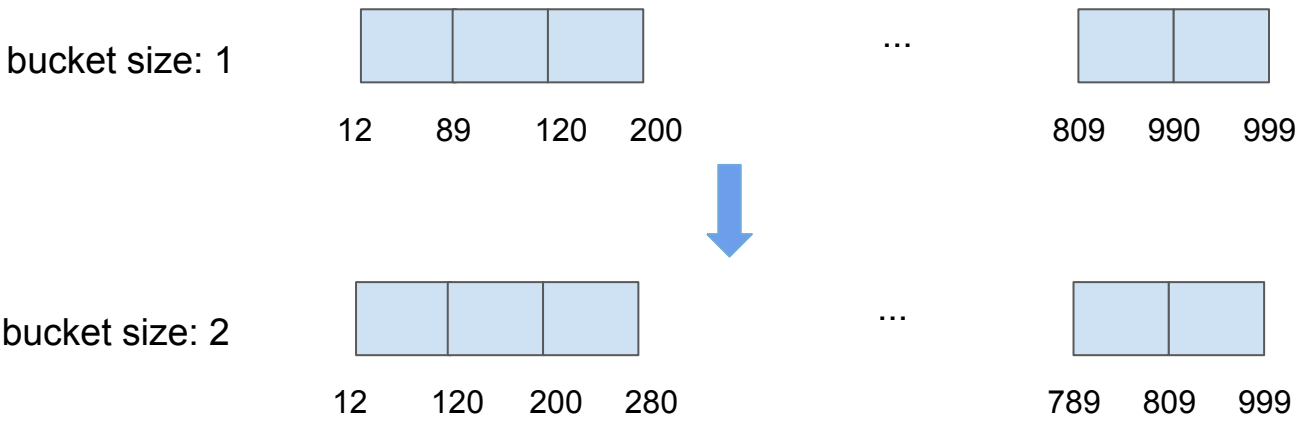


Plan optimization

Physical plan

- Statistics，例子：select * from t where c1 < 10 and c2 < 10（c1 is PK，c2 is index）

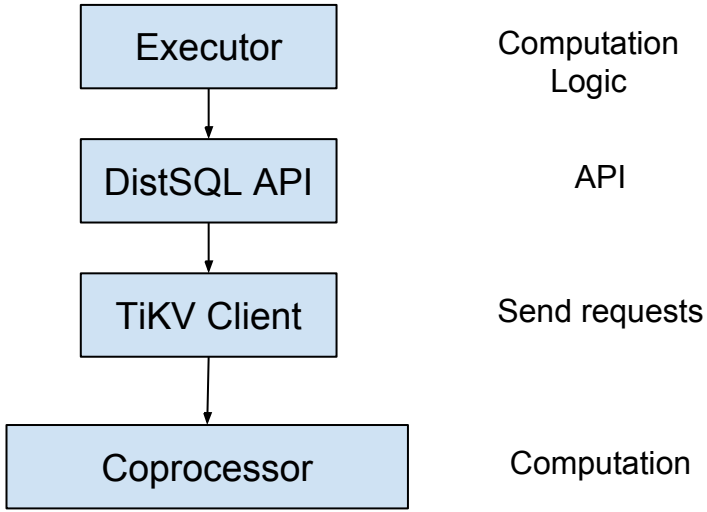
  等深直方图，bucket count 128-256



Dist SQL

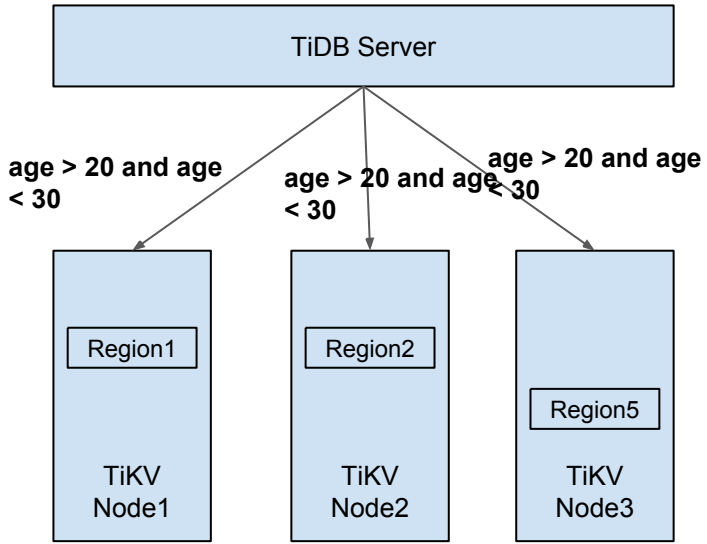
分布式计算

- 减少计算成本
- 减少网络开销



Dist SQL

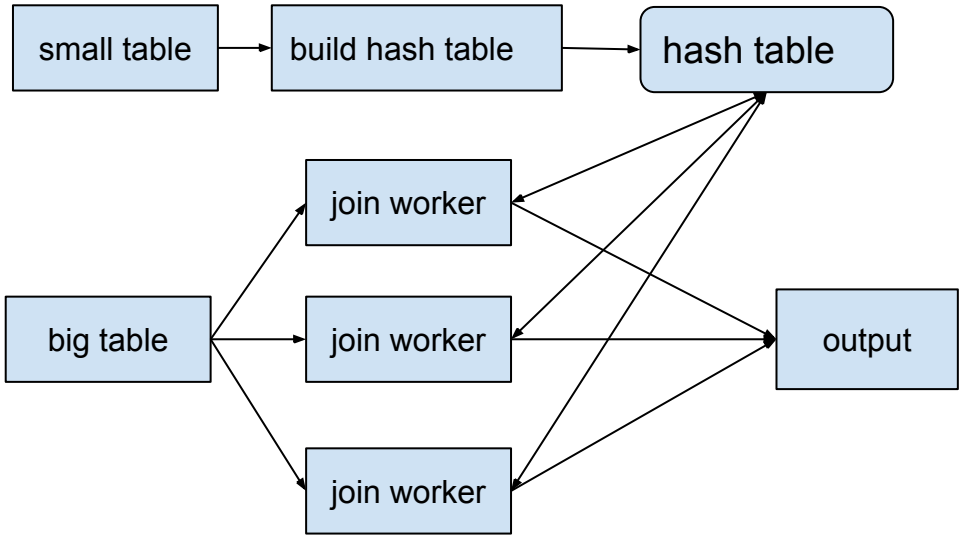
- select * from t where age > 20 and age < 30;
- select count(id) from t where age > 20 and age < 30;
- select * from t order by age limit 10;



Hash join

并行优化，支持 hash join

- 小表放到内存，等 值 key 建立哈希表
- 大表是用 goroutine 分批取值，匹配哈希表
- 之后会支持 merge sort join 等常用算法



Online DDL

现状，锁表（有些数据支持读操作，但是也以消耗大量内存为代价）

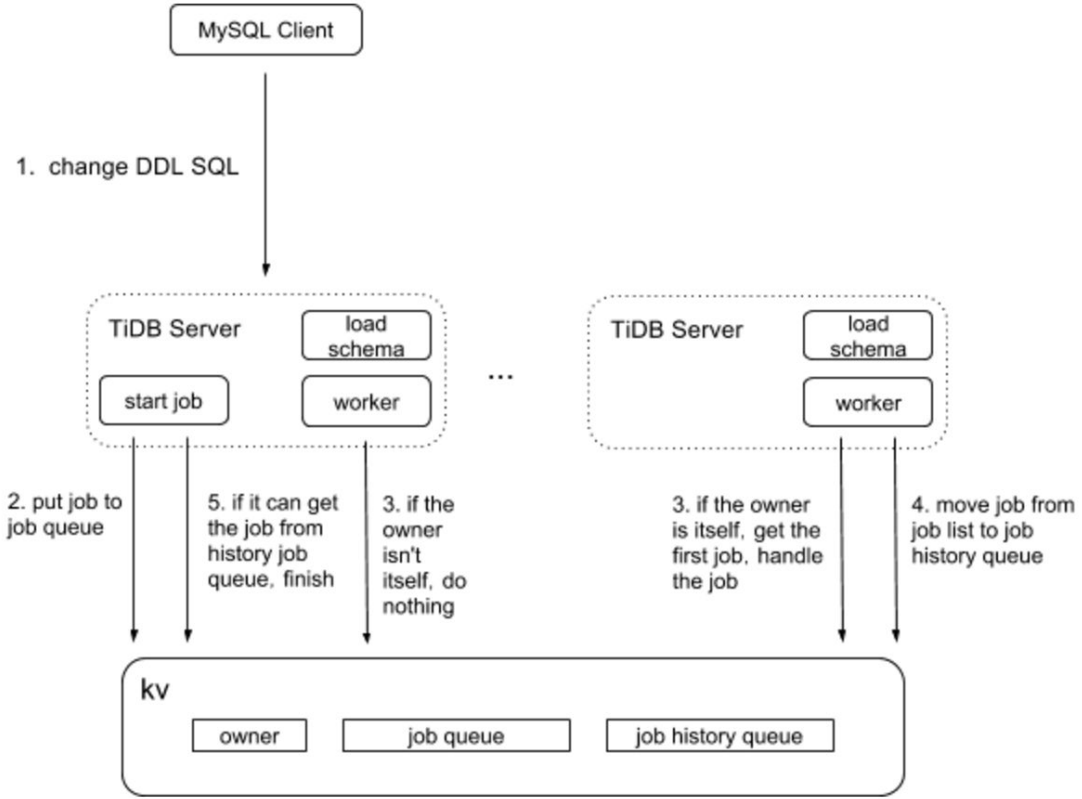
- 架构师们在设计整个系统的时候都会很慎重的考 虑表结构
- DBA 在做此类操作前要做足准 备

TiDB解决方案，参考 Google 动态变更 schema 的论文

```
absent --> delete only --> write only -- reorg --> public
```

Online DDL

- 一般 DDL 实现
- 特殊 DDL 实现
  - drop table/database



Online DDL

Add index 优化

同时并发地处理 defaultBatches(16) 个任务. 每个任务分别处理一个 handle 区间的 index 记录. 虽然每个 handle 区间的长度可控, 但是这个区间的 handle 值不可预计(handle 虽然是递增的, 但是里面的记录可能被删除过), 所以需要串行地获取 handle 区间. 即真正的并发处理需要在获取完 handle 区间之后执行.

Add column 优化

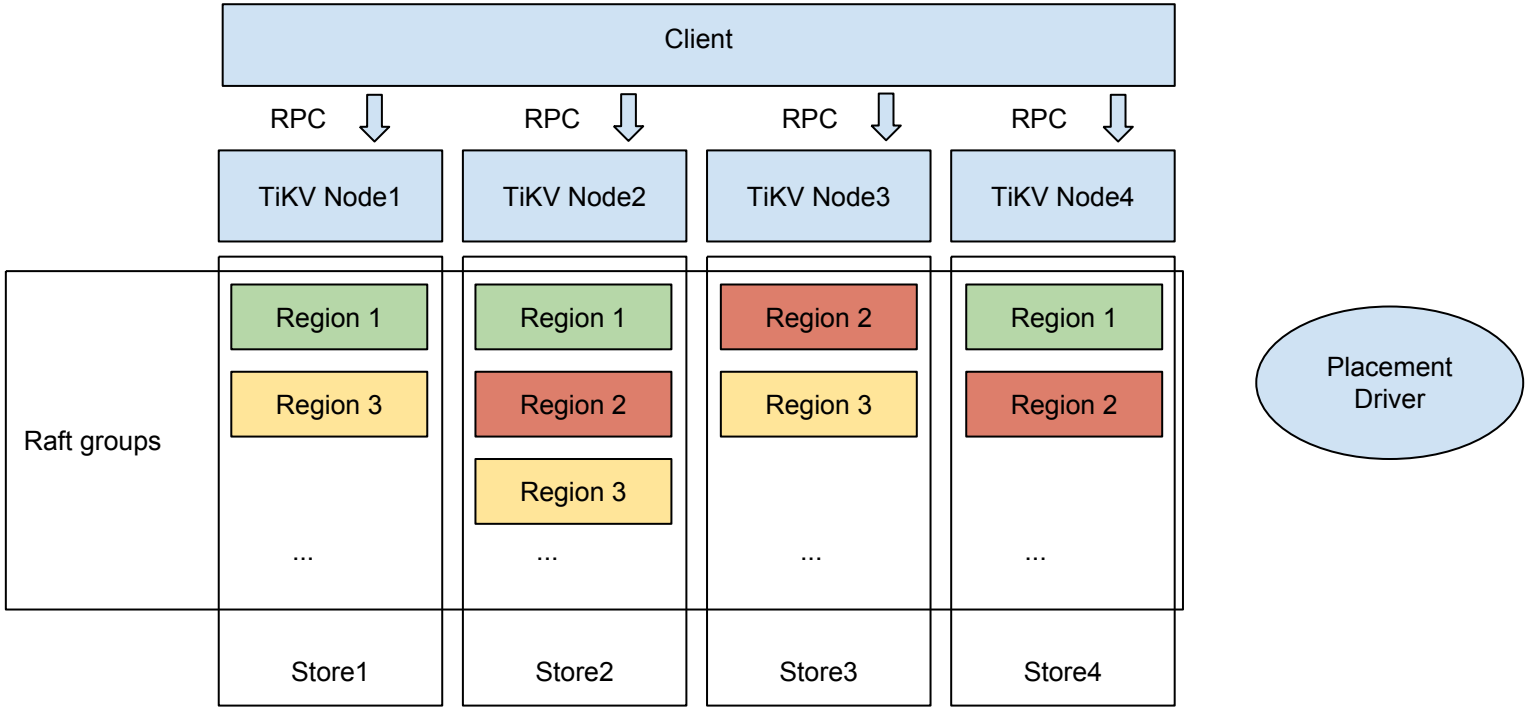
1. The default value is nil and the not nul flag is false
2. The default value isn't nil

TiDB

通过 TiKV 支持的特性:

- 分布式事务
  - 2PC(二阶段提交)，参考 Google percolator 的论文
  - MVCC
  - 隔离级别（SI + 乐观锁）
  - 引擎 RocksDB
- 水平扩容/缩容
  - raft 协议 + PlacementDriver
- 容错

TiKV


Client
RPC
RPC
RPC
RPC
TiKV Node1
TiKV Node2
TiKV Node3
TiKV Node4
Raft groups
Region 1
Region 3
...
Region 1
Region 2
Region 3
...
Region 2
Region 3
...
Region 1
Region 2
...
Store1
Store2
Store3
Store4
Placement Driver

Feelings

- 代码风格
  - 对于原先的我更多的是代 码命名方面的问题，一步一步的修改
  - 参照 Go 源码或者 GitHub 上一些知名 项目
  - 注释有时候是必要
- 测试
- 在这个团队里，让我的感受是没有挺好，你可以更好

Q & A

- 项目地址
- TiDB: http://github.com/pingcap/tidb （7000+ stars）
- TiKV: https://github.com/pingcap/tikv （1600+ stars）

PingCAP公众号