GOTC

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE , OPEN WORLD

分布式数据库与存储专场

本期议题：A History of TiDB Query Optimizer

张建

2021 年 08 月 01 日

About Me

GOTC

- 张建，开源爱好者，TiDB Maintainer
- TiDB SQL 团队负责人兼 TiKV 产品负责人@PingCAP
- 前 MaxCompute 执行引擎研发@Alibaba
- 关注数据库、分布式系统、开源软件与社区等领域
- Email: zhangjian@pingcap.com

Agenda

GOTC

01 What's TiDB

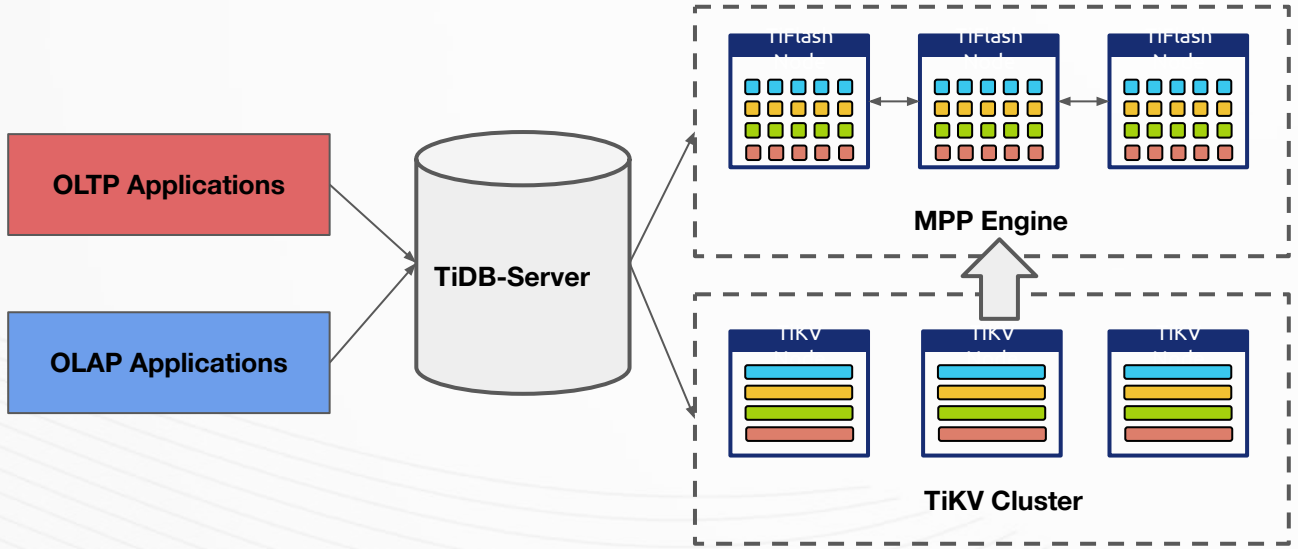
02 The History of TiDB Optimizer

03 Future Work

GOTC

What's TiDB

What's TiDB



TiDB：全新一栈式实时 HTAP 数据库

- 一键水平扩容或者缩容
- 金融级高可用
- 实时 HTAP
- 云原生的分布式数据库
- 兼容 MySQL 5.7 协议和 MySQL 生态

助推全球 1500+ 企业互联网化和数字化

GOTC

10+ 银行与头部金融企业

- 网联交易、网贷核算
- 互联网业务中台
- 实时风控、反欺诈

三大运营商

- 支付与风控
- 征信与账单系统
- 精准营销

覆盖中国互联网独角兽企业的 80%

- 万亿量级数据毫秒响应
- 支付、秒杀等场景
- 单个集群1000+ 节点

物流与零售

- 实时统计报表
- 数据中台
- 交易与报表业务隔离

美国、日本、东南亚头部支付企业与电商

- 在线支付
- 用户与订单管理
- 风控与营销

OLTP 规模化的头部用户

GOTC

面向零售高增长交易场景

全球领先餐饮巨头

- **实时高并发**，高效支撑短时高峰交易及数据访问，**1.5 亿用户**
- **弹性高扩展**，TiDB on K8s 随时随地在线扩展
- **安全高可用**，可靠放心的高可用容灾保障能力

在线核心联机支付交易平台

日本头部金融支付企业

- 在线支付联机交易，钱包等**高增长交易支撑，3000 万用户**
- **水平弹性扩展**和几乎线性的扩展能力
- **友好的开发界面**，应用几乎无需改动

面向金融敏态交易场景

平安人寿在线金融联机服务

- 支撑平安人寿“金管家”单日交易额破**1000 亿，超 1 亿用户**
- **高可靠、低延迟、可快速扩展**
- **大大降低敏态应用开发复杂度**，加快应用上线速度
- **在线弹性扩缩容**满足不确定业务需求

TiDB HTAP：一栈式数据服务生态

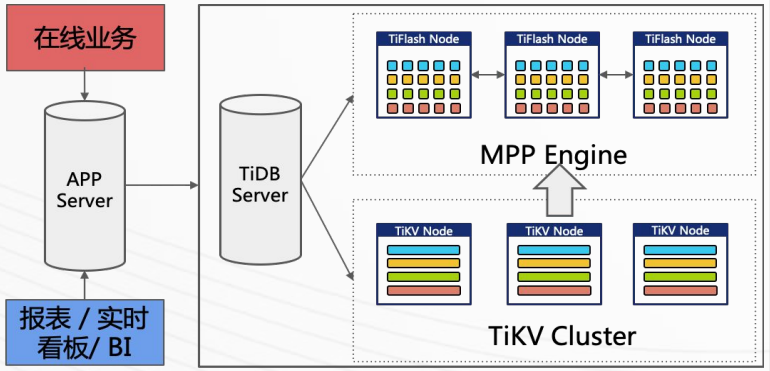
GOTC

混合负载场景

汽车之家 看车·买车·用车

交易处理和在线分析同步规模化

报表业务不影响在线交易

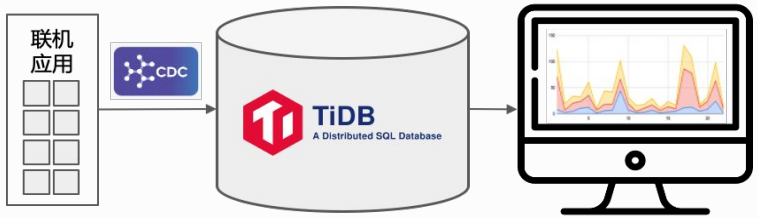


流式计算场景

360 www.360.cn

实时到账、实时统计

缩短业务等待周期，提升用户体验

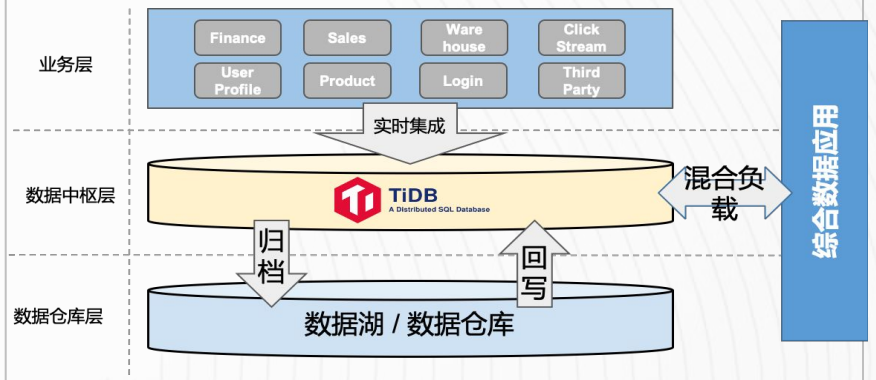


数据中枢场景

贝壳 找房大平台

多源汇聚，统一数据服务

节省系统和数据聚合代码开发时间



写 SQL 就行了

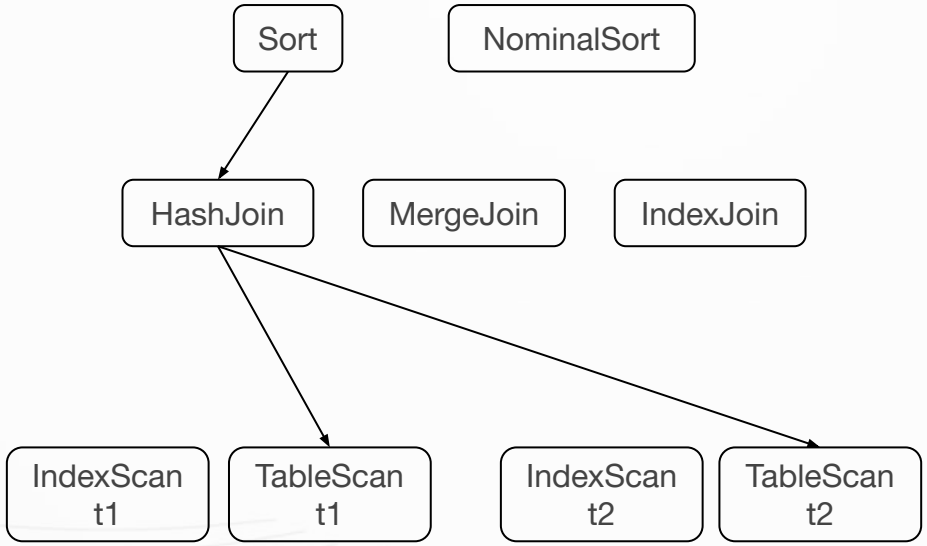
一个TiDB 系统，一个访问入口，一份数据

GOTC

The History of TiDB Optimizer

What's Query Optimizer

```
select t1.a
from t1
join t2
on t1.col1 = t2.col1
order by t1.col1
```



目标:做一个执行计划

- 输入 AST
- 输出物理执行计划

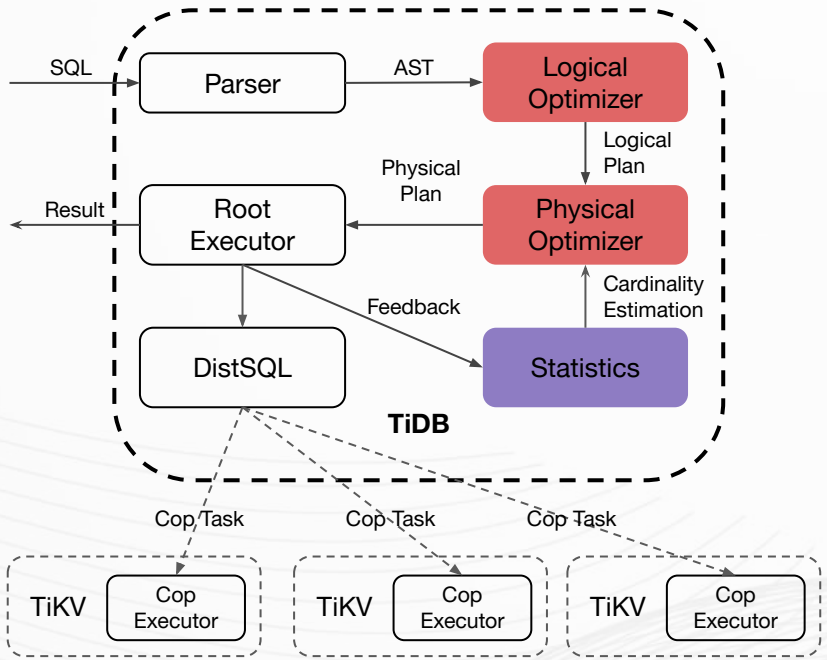
物理执行计划需要考虑什么内容:

- 选哪个索引
- 子查询如何执行
- 用什么 Join 算法和 Join 顺序
- ...

TiDB 优化器框架简介

GOTC

采用 1979 年由 Selinger 在 System R 中使用的经典优化器模型



优化器的关键要素：

- 搜索空间
- 搜索算法
- 代价估算

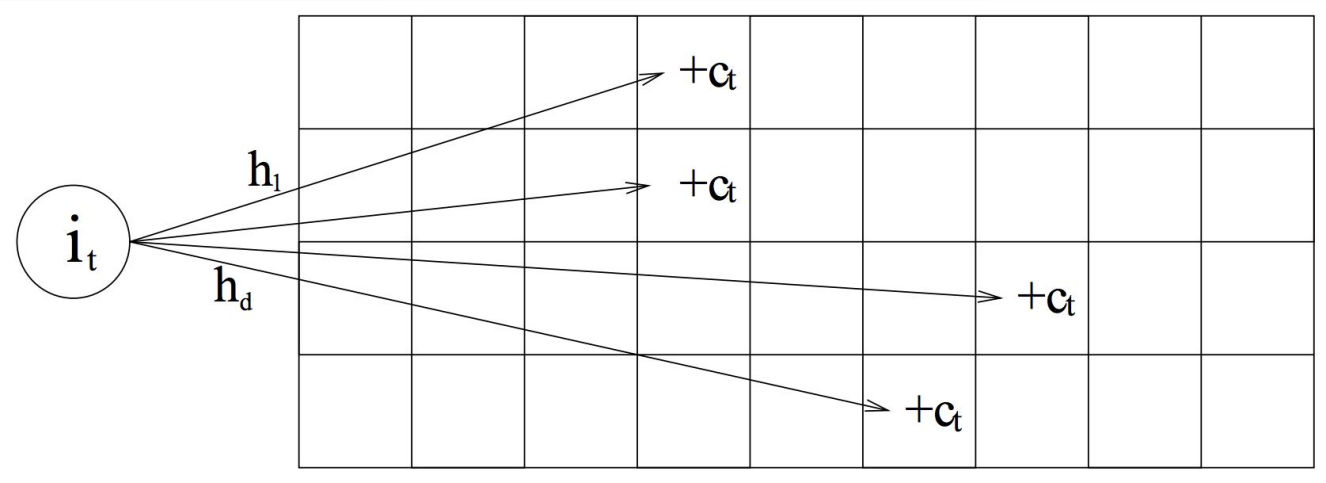
统计信息简介

统计信息能帮我们做什么

- 有多少数据满足条件：id between 10129 and 23819
- 按照索引 date 去读取到第一个 id > 10129 的数据要扫多少数据：where id > 10129 order by date limit 1
- 有多少数据满足条件：s join t on s.id = t.id
- ...

初始阶段:拥有基本的统计信息和 CBO 能力

TiDB 1.0 中的统计信息简介:



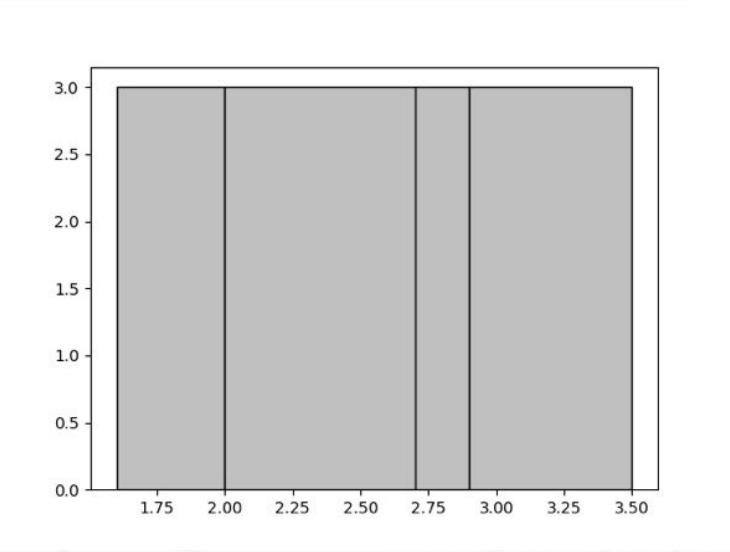
Count-Min Sketch

where date = "2021-08-01"

where status **in** ("Desiging", "testing")

初始阶段:拥有基本的统计信息和 CBO 能力

TiDB 1.0 中的统计信息简介:



Equal-Depth Histogram

where date **between** "2021-07-31" **and** "2021-08-01"

where age > 20;

遇到的问题

- 在线业务忽然卡了
- 集群中出现了好多慢查询
- TiDB OOM 了
- ...

先看执行计划

GOTC

```
TiDB > EXPLAIN SELECT a.col2, a.col3, a.col4, a.col5 FROM tblA a, tblB b WHERE a.col1=b.col2 AND b.col1=4;
+---------------+--------------+-----------------------------------+------+-----------------------------------------------------------------+-------+
| id            | parents      | children                          | task | operator info                                                   | count |
+---------------+--------------+-----------------------------------+------+-----------------------------------------------------------------+-------+
| TableScan_10  | Selection_11 |                                   | cop  | table:b, range:(-inf,+inf), keep order:false                    |   0.8 |
| Selection_11  |              | TableScan_10                      | cop  | eq(b.col1, 4)                                                   |   0.8 |
| TableReader_12 | IndexJoin_6 |                                   | root | data:Selection_11                                               |   0.8 |
| TableScan_9   |              |                                   | cop  | table:a, range:(-inf,+inf), keep order:true                     |     1 |
| TableReader_13 | IndexJoin_6 |                                   | root | data:TableScan_9                                                |     1 |
| IndexJoin_6   | Projection_5 | TableReader_12,TableReader_13     | root | outer:TableReader_13, outer key:b.col2, inner key:a.col1        |   0.8 |
| Projection_5  |              | IndexJoin_6                       | root | a.col2, a.col3, a.col4, a.col5                                  |   0.8 |
+---------------+--------------+-----------------------------------+------+-----------------------------------------------------------------+-------+
```

根据 parent 和 children 来判断数据在计算过程中的流向

提升执行计划可读性

GOTC

```
mysql> EXPLAIN SELECT /*+ HASH_JOIN(t1, t2) */ * FROM t1, t2 WHERE t1.id = t2.id;
+------------------------------+----------+-----------+-------------------------+-----------------------------------------------+
| id                           | estRows  | task      | access object           | operator info                                 |
+------------------------------+----------+-----------+-------------------------+-----------------------------------------------+
| HashJoin_30                  | 12487.50 | root      |                         | inner join, equal:[eq(test.t1.id, test.t2.id)] |
| ├─IndexReader_35(Build)      | 9990.00  | root      |                         | index:IndexFullScan_34                        |
| │ └─IndexFullScan_34         | 9990.00  | cop[tikv] | table:t2, index:id(id)  | keep order:false, stats:pseudo                |
| └─IndexReader_33(Probe)      | 9990.00  | root      |                         | index:IndexFullScan_32                        |
|   └─IndexFullScan_32         | 9990.00  | cop[tikv] | table:t1, index:id(id)  | keep order:false, stats:pseudo                |
+------------------------------+----------+-----------+-------------------------+-----------------------------------------------+
5 rows in set (0.01 sec)
```

使数据依赖关系一目了然：

- 树形结构：上层的节点依赖下层的节点数据作为输入
- Build 和 Probe 两个关键：Probe 依赖 Build 作为输入

丰富各种情况下问题排查的方式

希望知道实际运行时哪里慢了，优化器估算值和实际值的差距：**explain analyze**

一个 SQL 长时间运行不结束，希望知道它的执行计划是什么，卡在哪里：**explain for connection**

希望知道慢查询日志中的 SQL 为什么慢：从记录 explain 结果到**记录 explain analyze 结果**

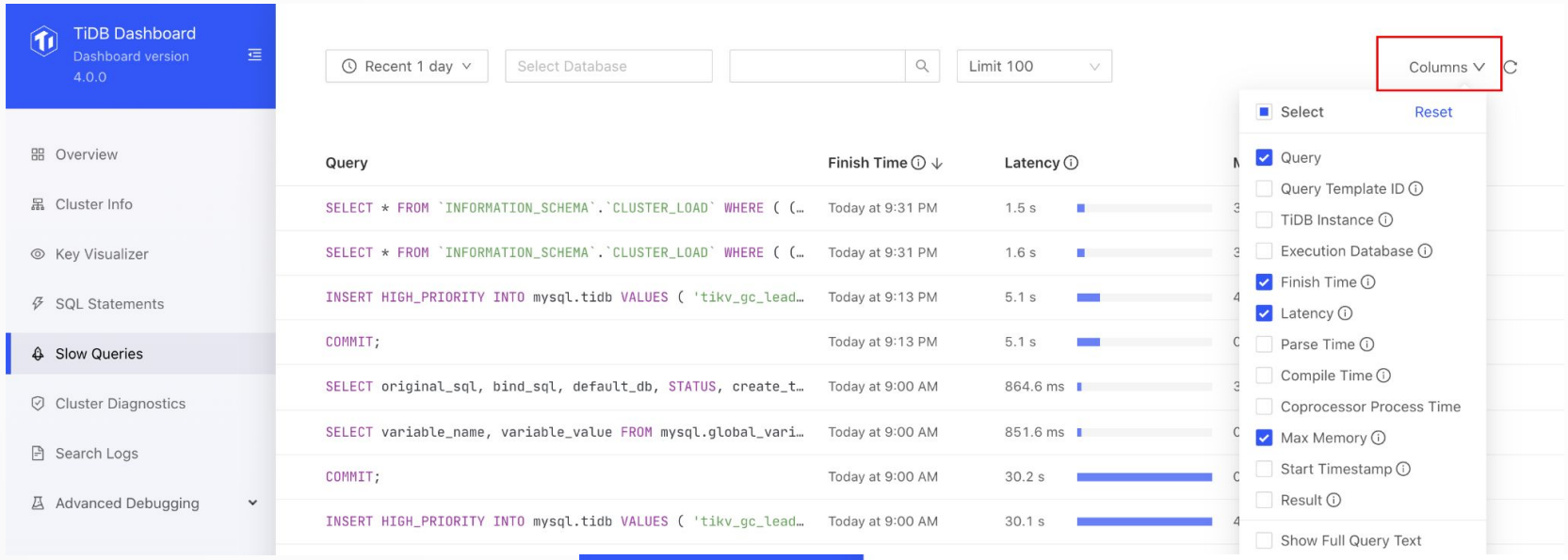
希望知道这个 SQL 历史上的执行计划是什么，执行时间分别什么样：**statement summary 系列的内部表**

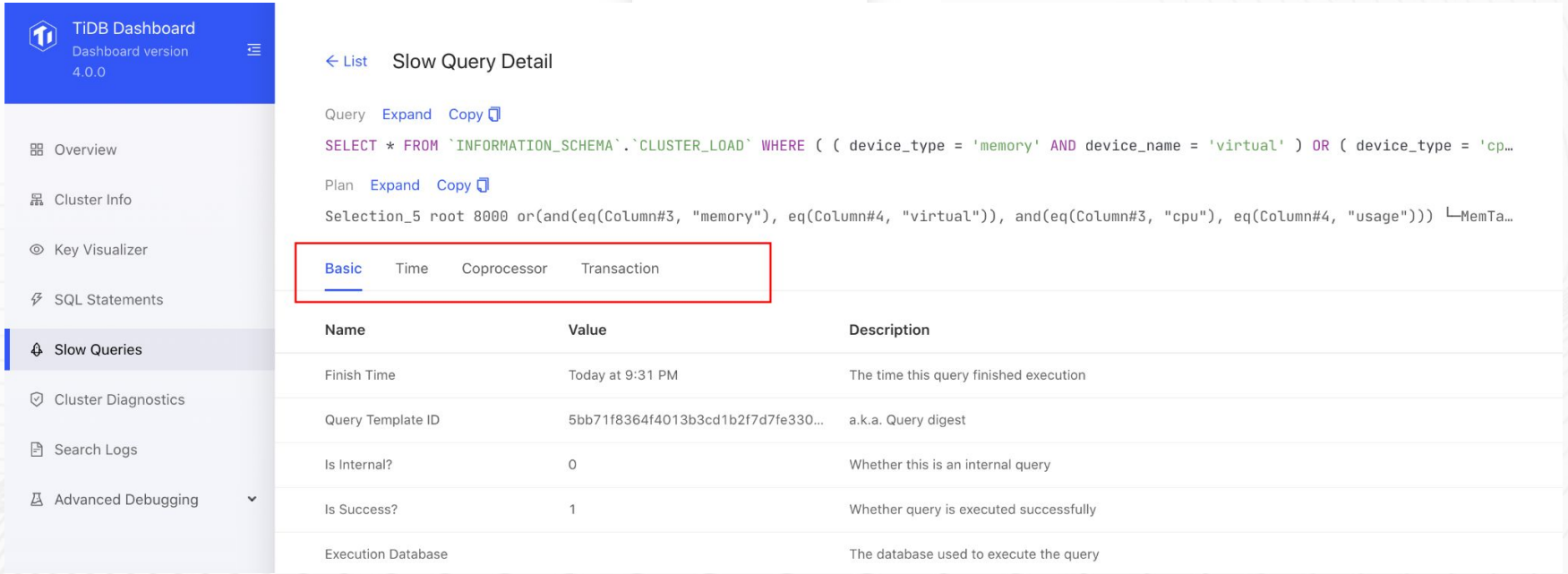
希望一眼看出哪个 SQL 慢，慢在哪里：**TiDB Dashboard**

趋势：信息越来越丰富，工具越来越易用

TiDB Dashboard

GOTC





先加 Hint，提供 workaround 方法

什么是优化器 Hint：

- 告诉优化器选哪个具体的索引
- 告诉优化器用什么 Join 算法
- 告诉优化器用什么聚合算法

```
SELECT /*+ USE_INDEX(t1, idx1), HASH_AGG(), HASH_JOIN(t1) */ count(*)
FROM t t1, t t2 WHERE t1.a = t2.b;
```

Hint 有哪些问题

给 SQL 加 Hint 需要改业务 SQL：

- SQL 散落在各个地方不好改
- 改 SQL 涉及到应用代码的修改，需要完整走一遍应用程序的发布流程
- 有些 SQL 是应用程序引用的第三方库发的，没法改

从 Hint 到 SQL Binding

GOTC

SQL Binding：

- 不用修改 Hint
- 需要 super 权限

```
CREATE BINDING FOR

SELECT count(*)
from t t1, t t2 WHERE t1.a = t2.a;

USING

SELECT /*+ USE_INDEX(t1, idx1), HASH_AGG(), HASH_JOIN(t1) */ count(*)
FROM t t1, t t2 WHERE t1.a = t2.b;
```

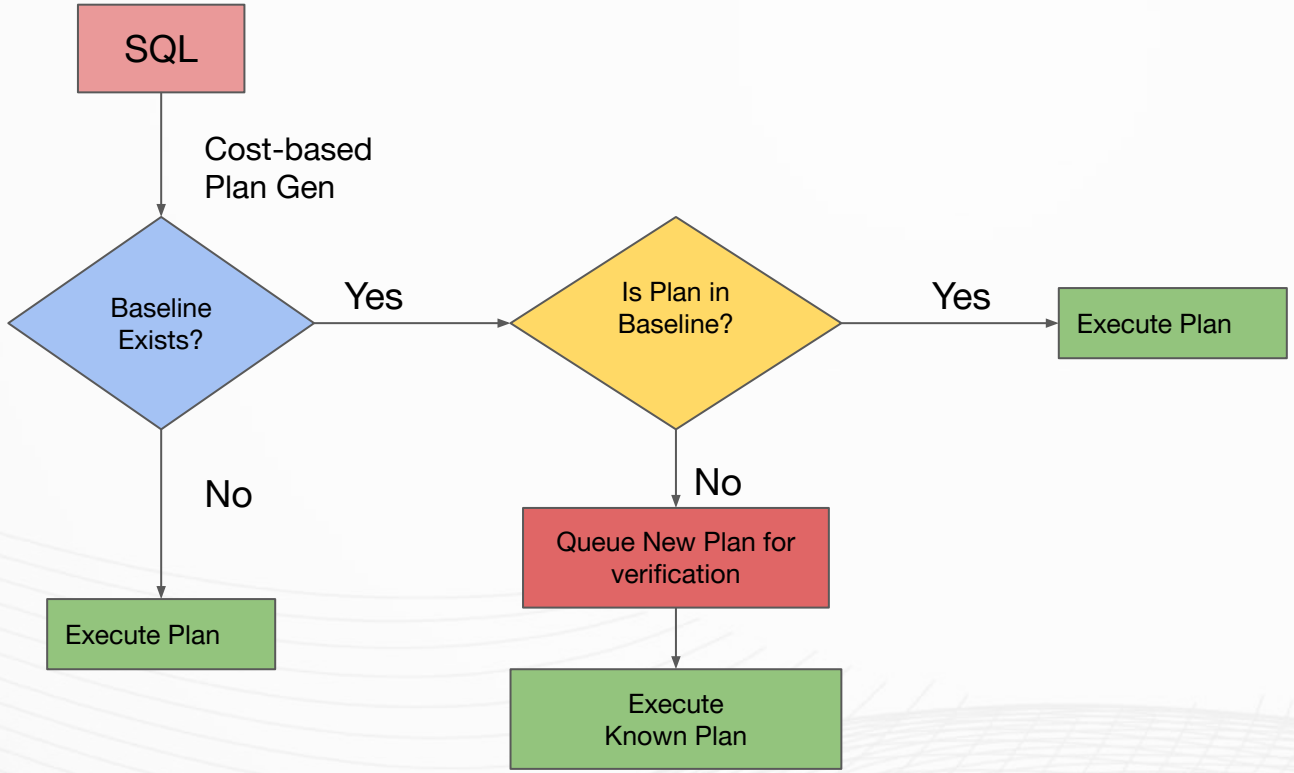
SQL Binding 有哪些问题？

“我有很多 SQL，成千上万，一个个加 Binding 太累了！”

“万一 SQL 参数的值变化了，原先的执行计划不够好怎么办？”

```
select * from t
where date between ? and ?;
```

从 SQL Binding 到 SPM



SPM 两大特性：

- 自动捕获
- 自动演进

于此同时，我们也在完善优化规则和统计信息

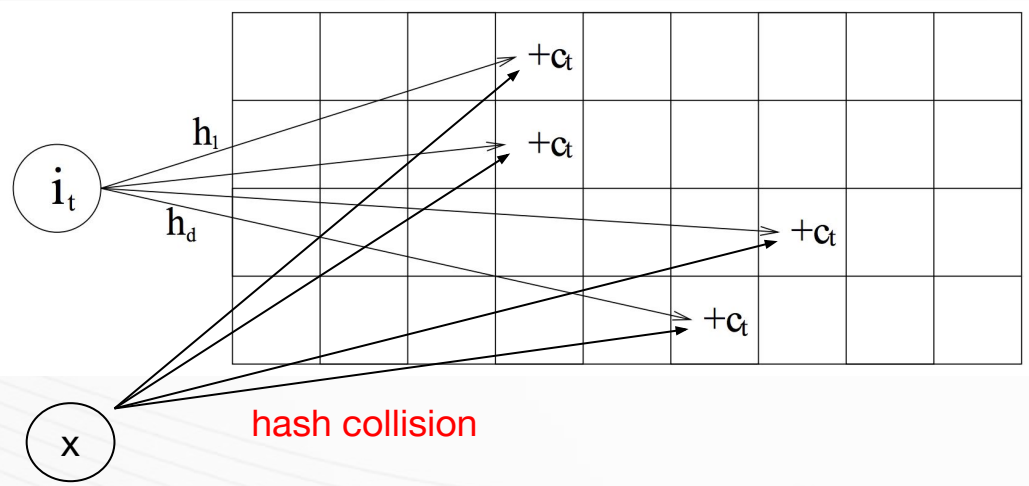
```
CREATE TABLE t(
    id INT, date DATETIME,
    INDEX idx1(id),
    INDEX idx2(id, date)
);

SELECT * FROM t
WHERE id = 10291
  AND date = "2021-08-01";
```

缩小搜索空间，降低犯错的概率：

- 引入 Skyline Pruning 裁剪一定不优索引
- 引入更多一定优的启发式规则

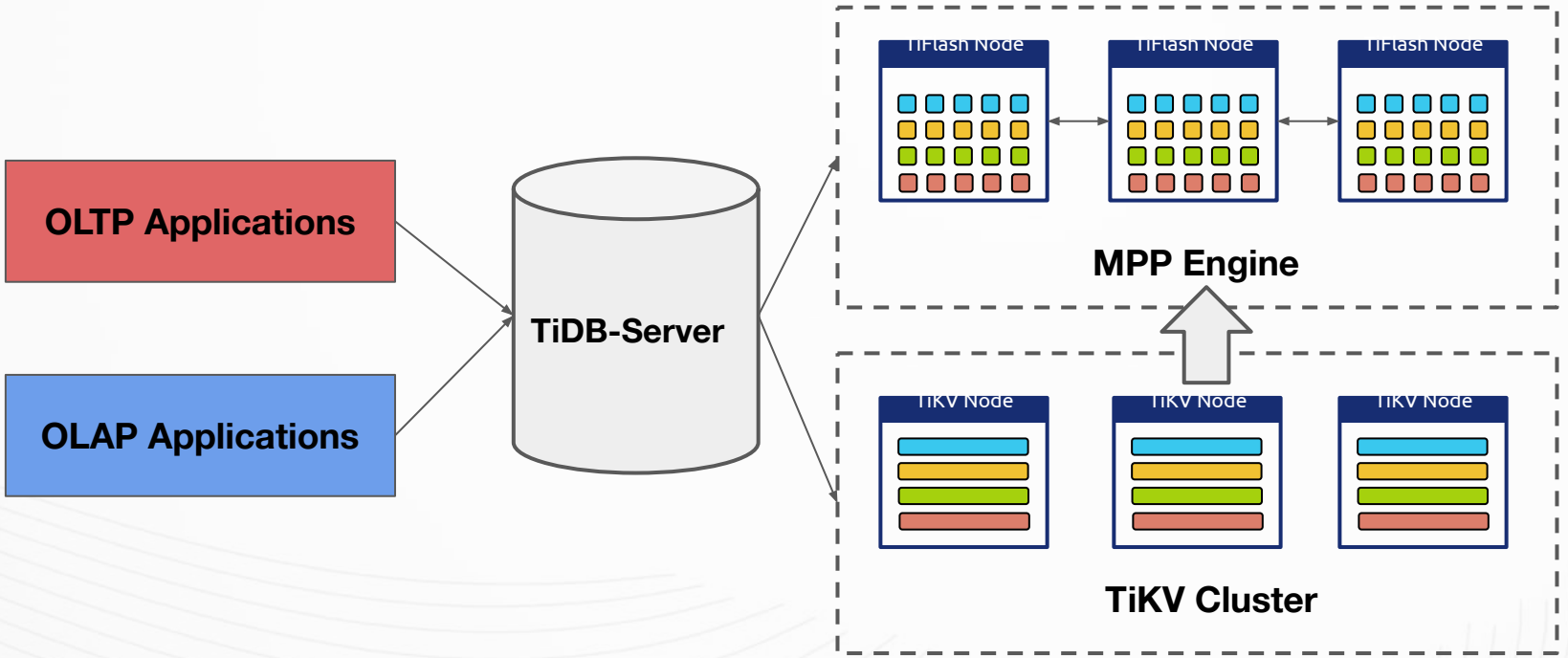
于此同时，我们也在完善优化规则和统计信息



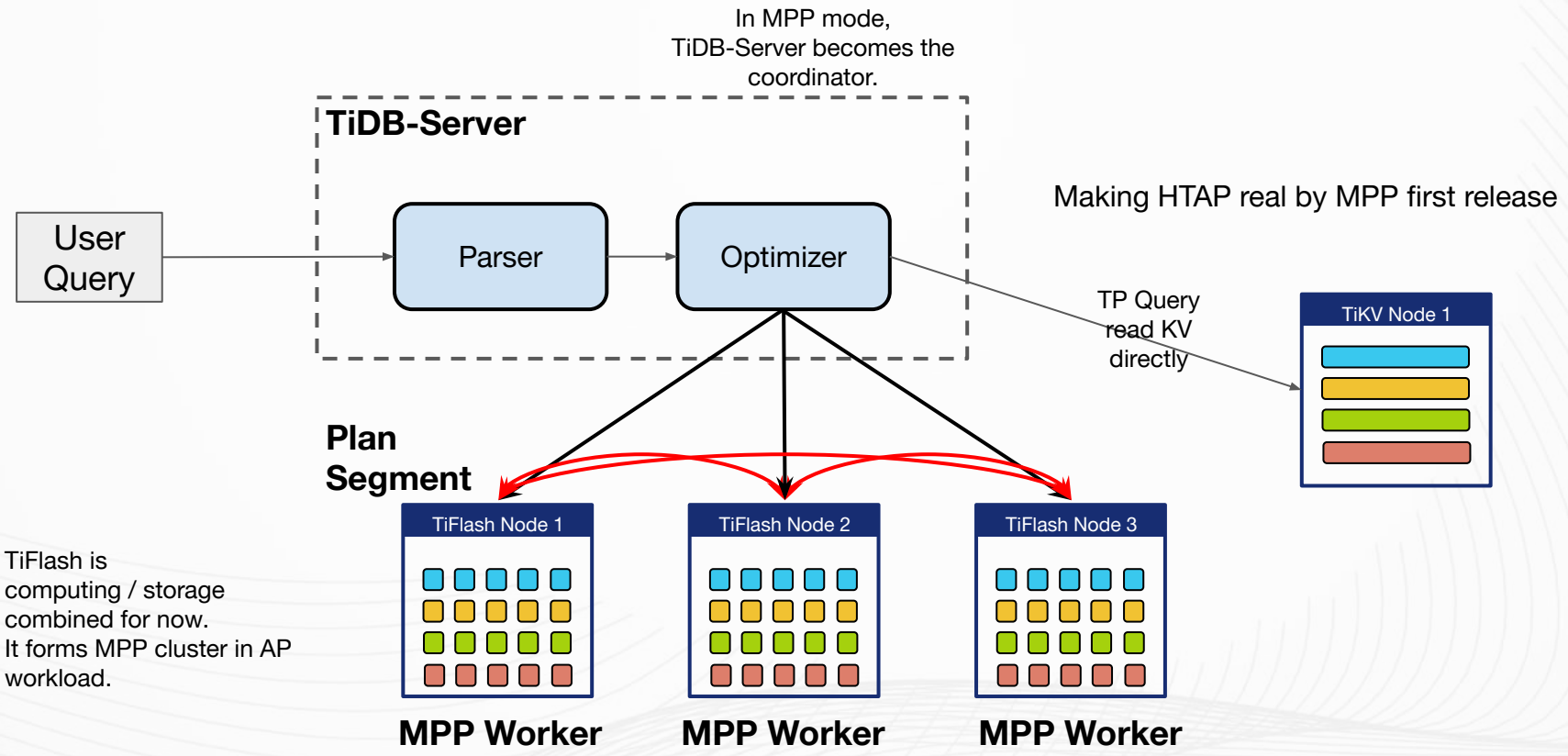
降低基数估算误差，提升选对执行计划的能力：

- 引入 TopN 的统计信息
- 引入相关系数的统计信息
- 去除 CM-Sketch 的统计信息

从 OLTP 优化器成长为 HTAP 优化器



HTAP 的优化器架构

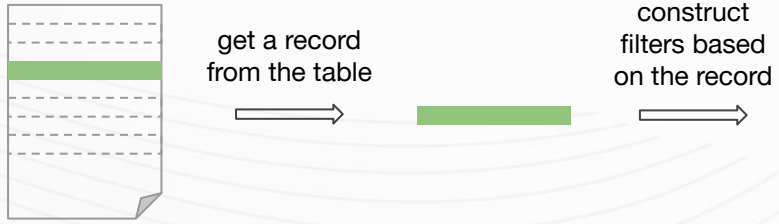

In MPP mode,
TiDB-Server becomes the
coordinator.
TiDB-Server
User
Query
Parser
Optimizer
Making HTAP real by MPP first release
TP Query
read KV
directly
TiKV Node 1
Plan
Segment
TiFlash Node 1
TiFlash Node 2
TiFlash Node 3
TiFlash is
computing / storage
combined for now.
It forms MPP cluster in AP
workload.
MPP Worker
MPP Worker
MPP Worker

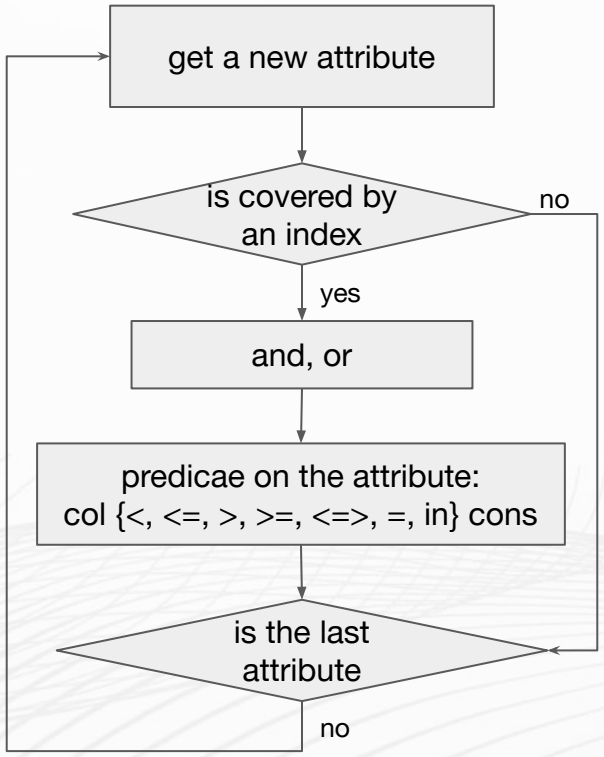
复杂的优化器系统，需要有科学的测试体系

GOTC

Optimizer Effectiveness

- 含义：优化器选中的执行计划不差于多少比例的执行计划
- 参考："OptMark: A Toolkit for Benchmarking Query Optimizers"





复杂的优化器系统，需要有科学的测试体系

GOTC

用 q-error 来衡量优化器的估算误差，降低它：

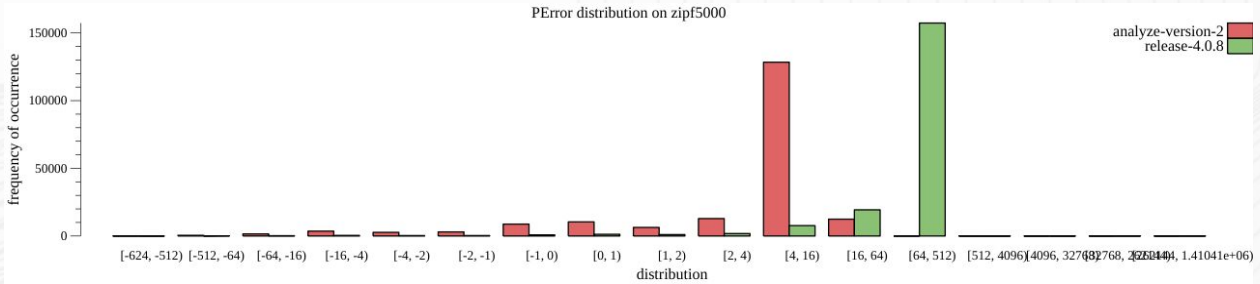
- 含义：max(est/act, act/est)
- 参考："Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors"

OverEstimation Statistics

Instance	Total	P50	P90	P99	Max
analyze-version-2	170394	14.040	14.040	17.800	70.400
release-4.0.8	188373	326.500	326.500	326.500	1410409.000

UnderEstimation Statistics

Instance	Total	P50	P90	P99	Max
analyze-version-2	20040	-1.460	-16.221	-107.000	-546.000
release-4.0.8	2062	-1.650	-21.064	-139.055	-624.576



总结

- TiDB 优化器架构
- 故障诊断和恢复
- 统计信息优化
- 做分布式并行执行计划
- 建立科学的测试体系

Future Work

Future Work

- **提高可预测性**：添加更多启发式规则
  - 为什么没有选这个索引？
- **提高可观测性**：Optimizer Trace
  - 为什么选了这个索引？
- **提高可扩展性**：Cascades Planner

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE