

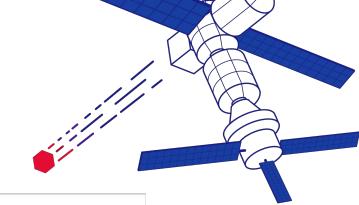


TiFlash 揭秘

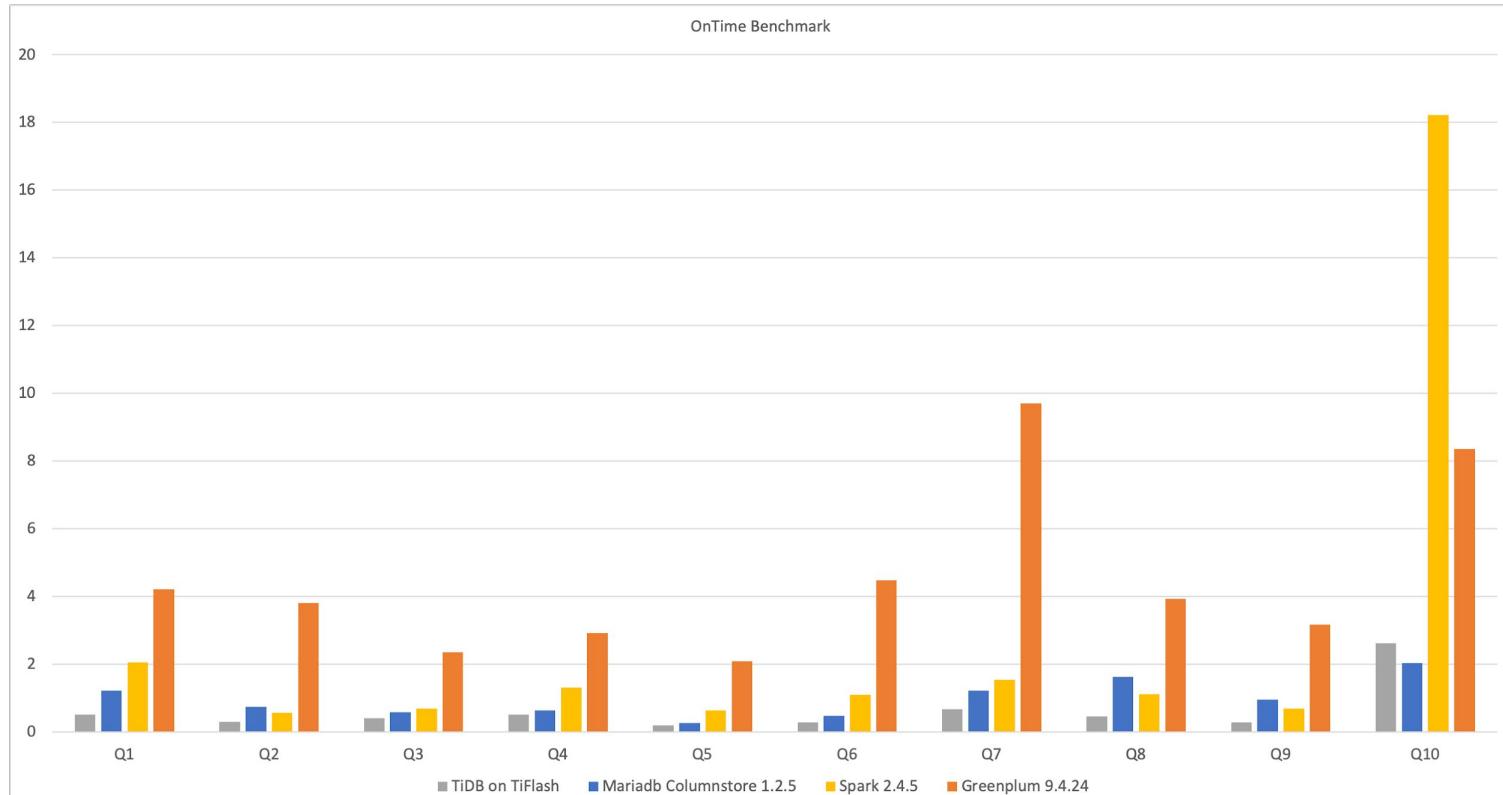


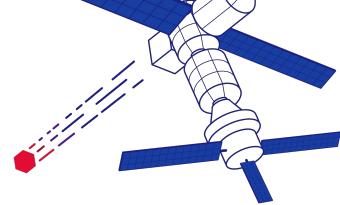
Benchmark!





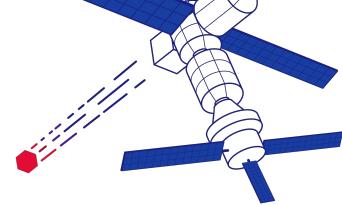
Benchmark OnTime





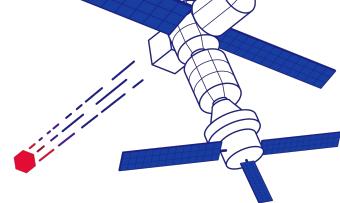
Benchmark OnTime

	TiDB + TiFlash	MySQL 5.7.29	Greenplum 6.1	Mariadb Columnstore 1.2.5	Spark 2.4.5 + Parquet	Oracle 12.2.0.1
Q1	0.508	290.340	4.206	1.209	2.044	88.53
Q2	0.295	262.650	3.795	0.740	0.564	76.05
Q3	0.395	247.260	2.339	0.583	0.684	74.76
Q4	0.512	254.960	2.923	0.625	1.306	74.75
Q5	0.184	242.530	2.077	0.258	0.627	67.44
Q6	0.273	288.290	4.471	0.462	1.084	134.08
Q7	0.659	514.700	9.698	1.213	1.536	147.06
Q8	0.453	487.890	3.927	1.629	1.099	165.35
Q9	0.277	261.820	3.160	0.951	0.681	76.5
Q10	2.615	407.360	8.344	2.020	18.219	127.29



我们测了什么？

- 借鉴自 ClickHouse 官网 ontime 的 11 范例条查询
- 1 / 3 表连接 + 聚合
- 剩余是宽表聚合(109 列)
- 这类查询囊括了多维分析的主要场景:根据不同维度筛选聚合来统计信息
- 为何我们只有 10 个测试结果?
 - 老实说我们偷偷隐藏了一个带有 count distinct 的查询结果, 因为看起来几家都不快, TiDB + TiFlash 也非常慢
 - count distinct 无法加速的改进将会在 3 月合入 TiDB 3.1 版本



我们测了什么？

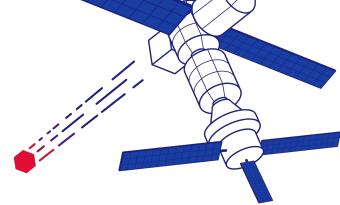
查询1：平均每月航班起降记录数

```
select avg(c1) from  
( select year, month, count(*) as c1 from ontime group by year, month ) A;
```

查询2：2000年到2008年的每日航班数

```
select dayofweek, count(*) as c from ontime  
where year>=2000 and year<=2008 group by dayofweek  
order by c desc;
```

以上这类多维分析聚合类查询大部分计算都可以由 TiFlash 加速，速度非常快



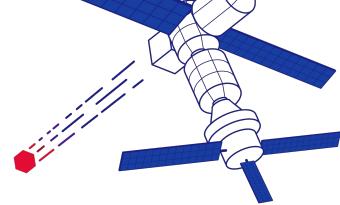
我们测了什么？

查询6:按照航空公司统计 2007 年延误比例

```
select carrier, c, c2, c*100/c2 as c3 from
(select carrier, count(*) as c from ontime where depdelay>10 and year=2007 group by
carrier ) A
inner join
( select carrier, count(*) as c2 from ontime where year=2007 group by carrier ) B using
(carrier)

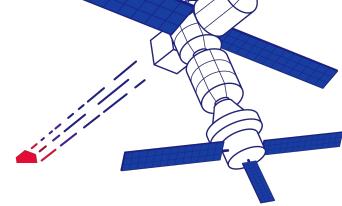
order by c3 desc;
```

Join 查询部分可以通过 TiDB 优化器规则进行聚合推过 Join 而享受 TiFlash



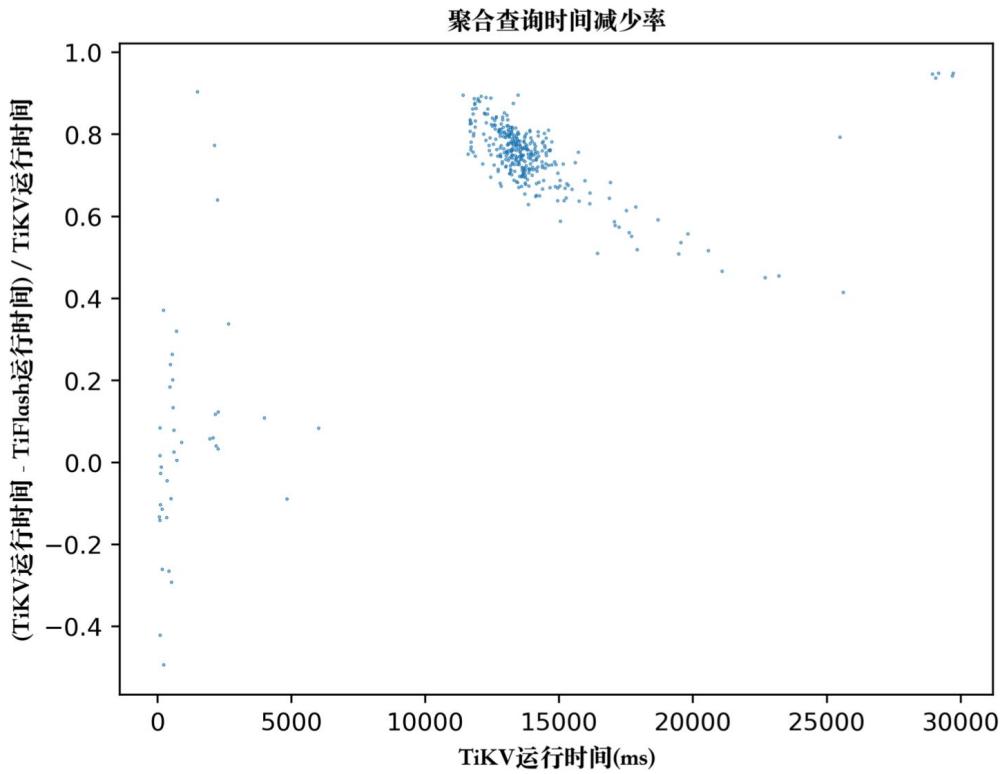
实际案例

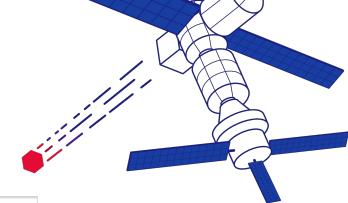
- 上海某大型社交电商公司案例(客户法务未审核暂时无法公布名字)
 - 之前已经在使用 TiDB 进行数据汇聚 + 实时分析
 - TiFlash 已经进入准生产, 等正式版发布会正式生产上线
 - 选择不增加机器而是和 TiKV 混合部署
 - 多维分析为主
 - 393 条不同查询普遍性能提升 3 - 4 倍
 - 长查询可以提速到 10 倍以上



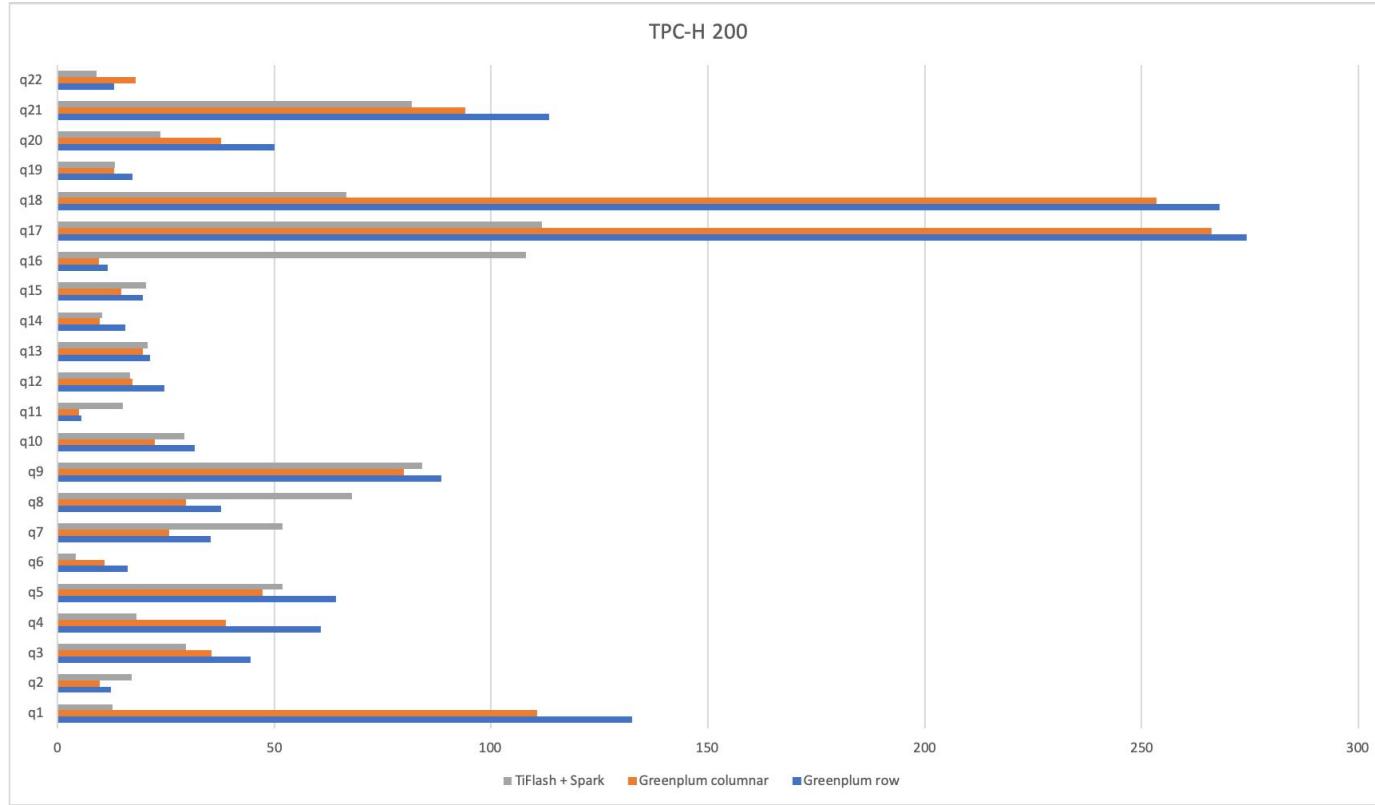
实际案例

来自客户使用报告



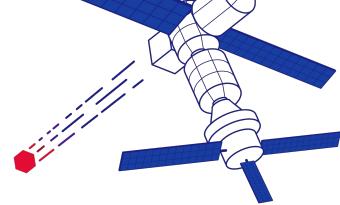


如果是更复杂的查询场景呢？



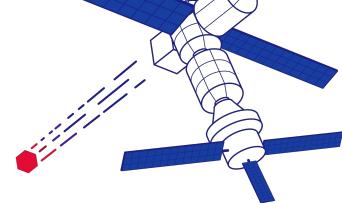
为何要做 TiFlash





| TiDB 分析场景之前有什么问题？

- 行存读取慢
 - 而列存才是分析场景的标配，配合向量化引擎如虎添翼
 - 行存在查询很少数据时对比列存有绝对优势
 - 但大量扫表且并非选中全部列时则 IO 效率极低
 - 后果是不管是 TiDB 还是 TiSpark 都无法达到满意的速度



TiDB 分析场景之前有什么问题？

行存

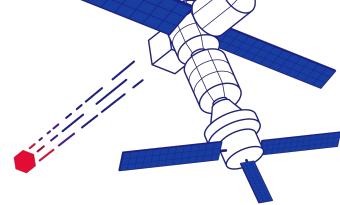
id	name	age
0962	Jane	30
7658	John	45
3589	Jim	20
5523	Susan	52

SELECT * from emp where id = 7658;

SELECT avg(age) from emp;

列存

id	name	age
0962	Jane	30
7658	John	45
3589	Jim	20
5523	Susan	52



| TiDB 分析场景之前有什么问题？

- 同一套 TiKV 无法隔离
 - 分析场景使用资源的方式倾向于同时投入尽可能多的资源以迅速完成计算
 - 分析场景存储方式也倾向于暴力扫描而非精确索引
 - 很多用户在进行 AP 查询时甚至需要很小心调整并发和读取速度，防止干扰，就算引擎可以算得快，也戴上了枷锁
 - TiSpark 无法盘活，极高的并发读取使得一旦开始计算那么整个集群无法安生

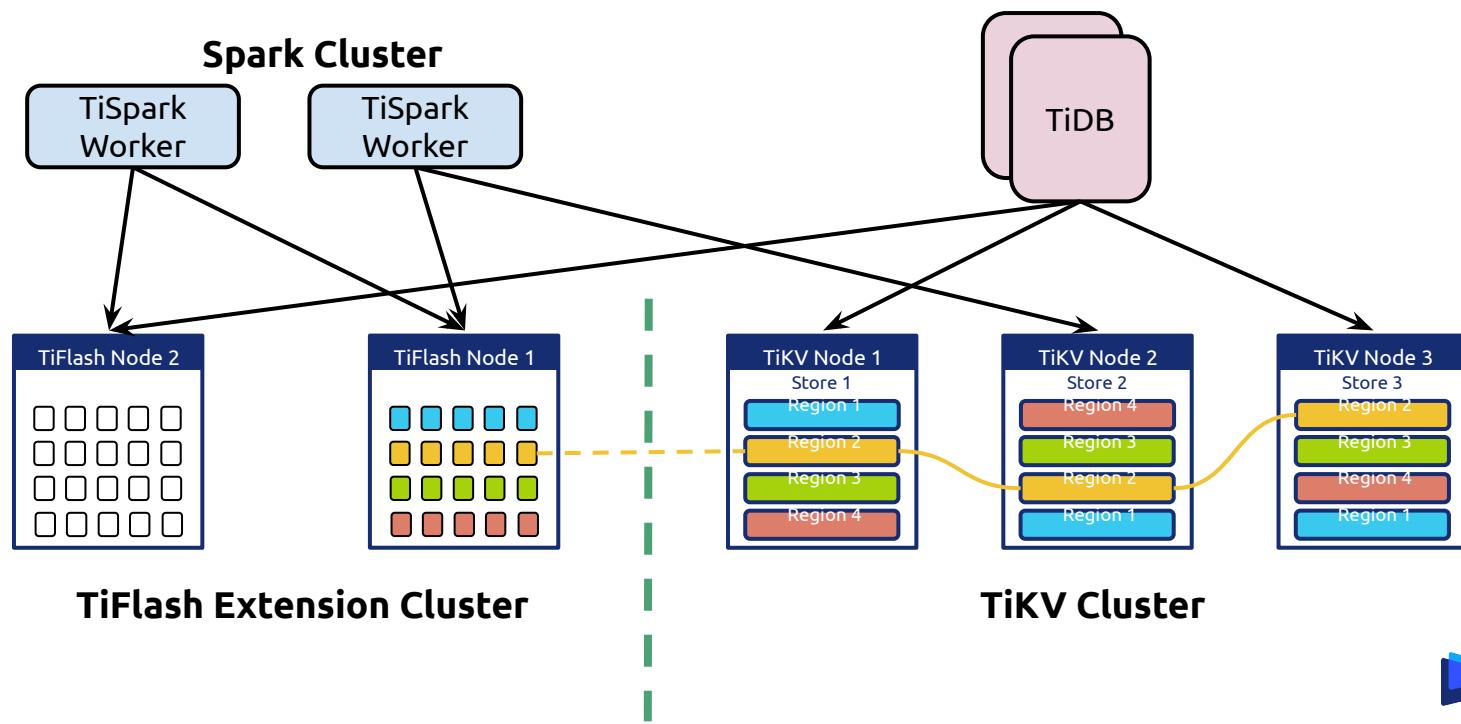
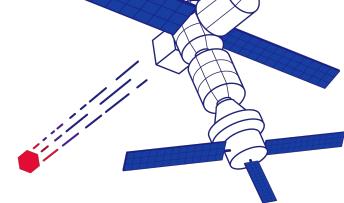
TP / AP 干扰

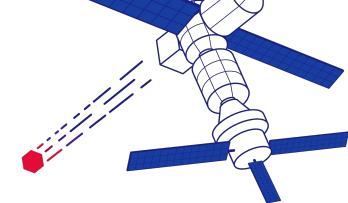


为何选 TiFlash

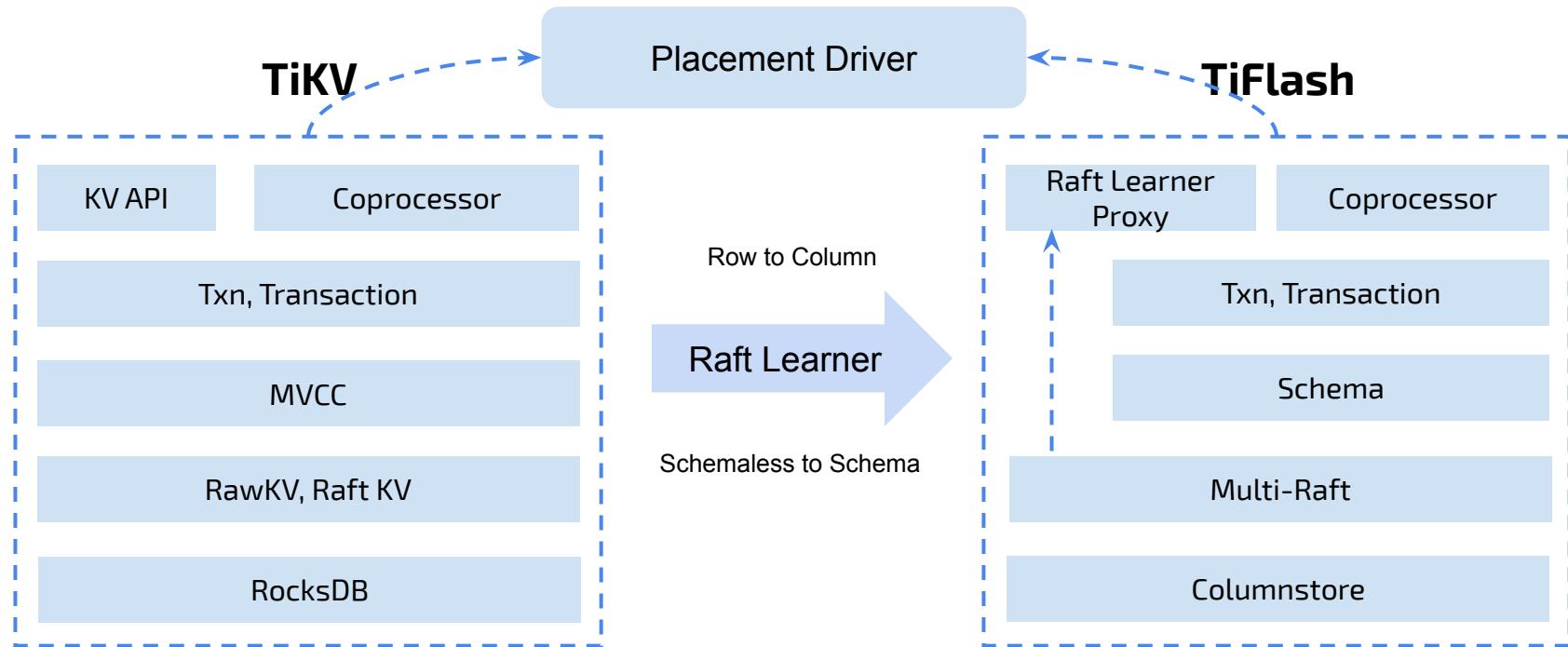


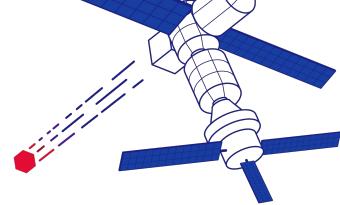
TiDB 3.1 新架构





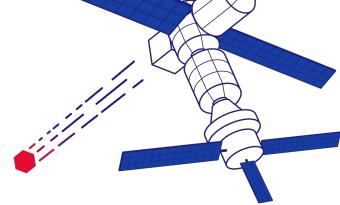
TiFlash 架构设计





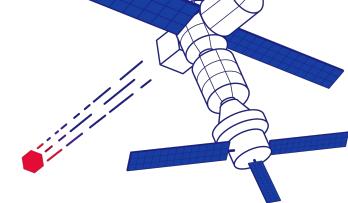
TiFlash 特色

- 高性能, 支持实时更新的列存引擎
 - 高速实时基于主键的 Update 支持
- 以 Learner 角色接入 Multi-Raft 体系
 - 负载均衡, 更快更方便扩展
 - 强一致的读取
- 配合 TP 场景使用: TP 数据实时高速可查无干扰
- 配合 AP 场景使用: 可自动大幅提升查询速度

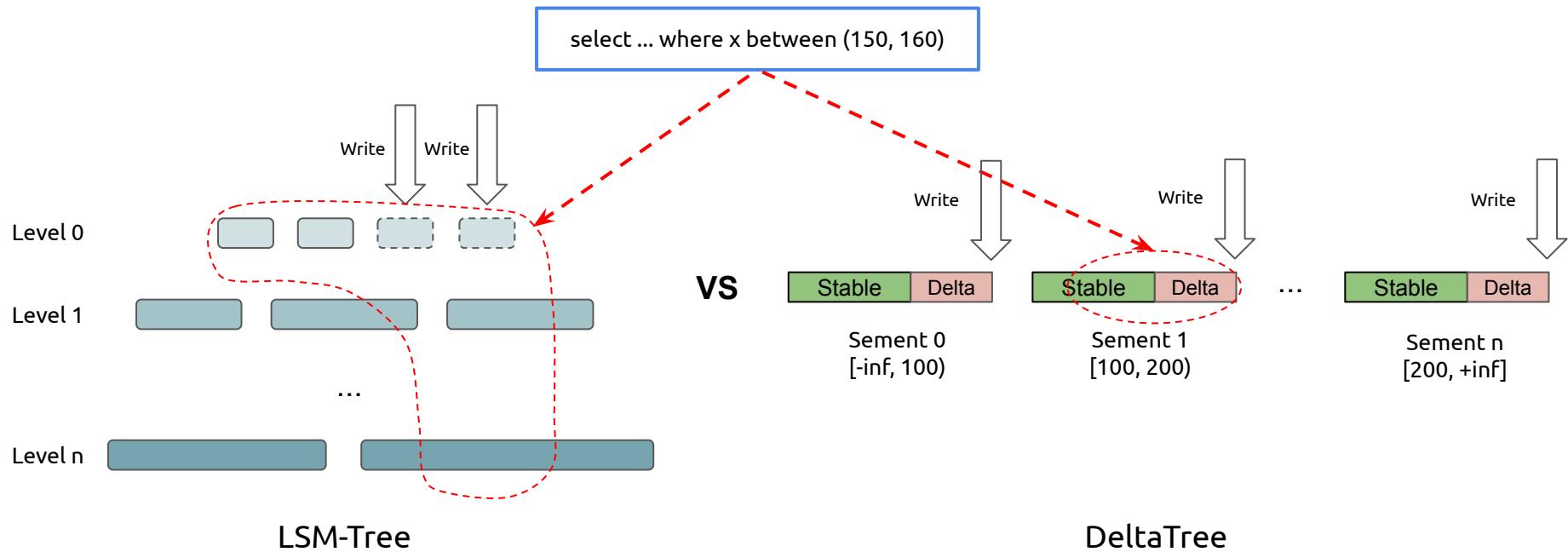


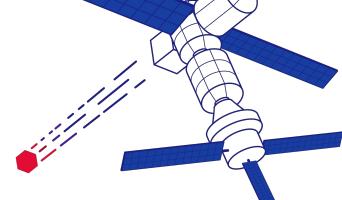
| 更快的查询响应

- 与 TiKV 不同的存储设计，更高效的批量读取性能
 - 分析场景特化的存储结构设计，更低的读取消耗
- 配合源于 ClickHouse 的极致向量化计算引擎
 - 更少的废指令，SIMD 加速



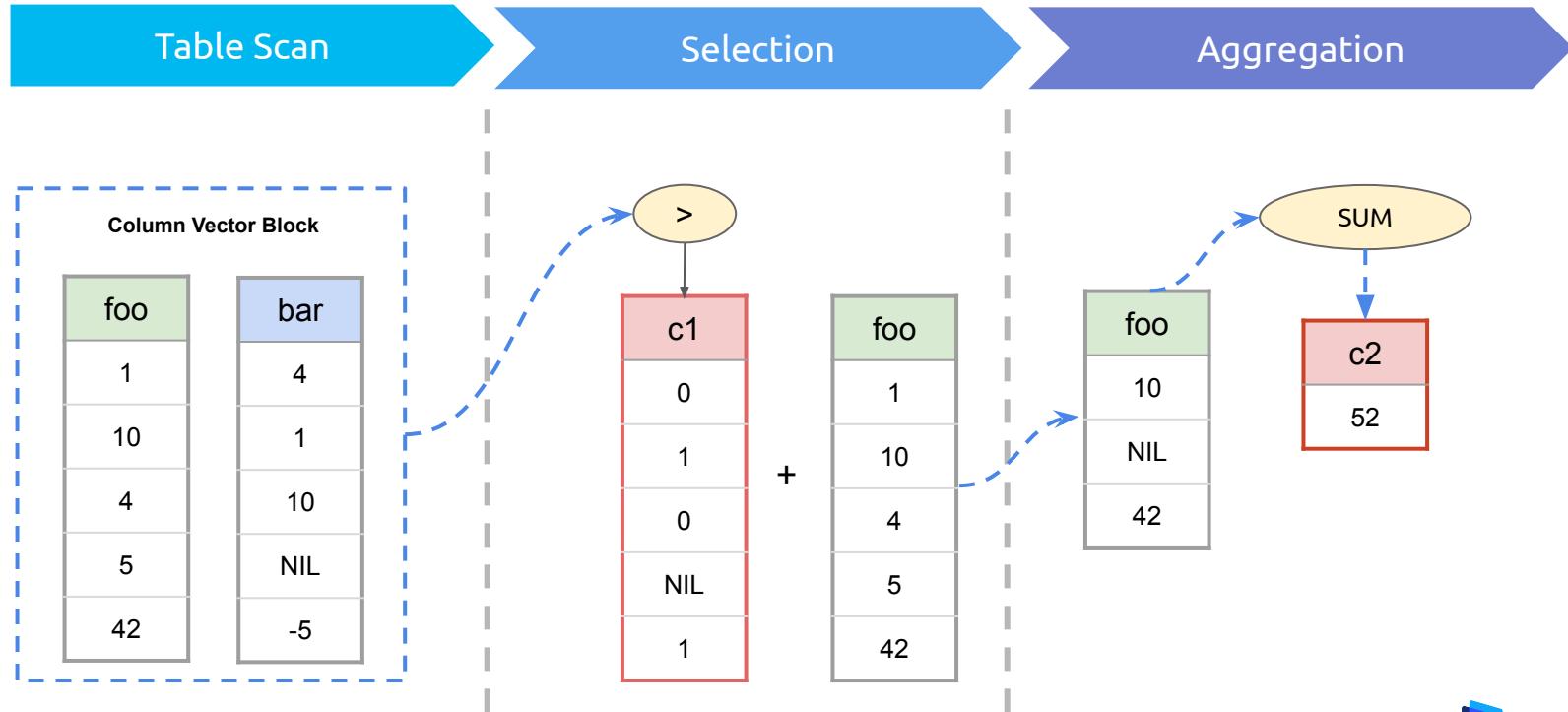
| 更快的查询响应 - 存储

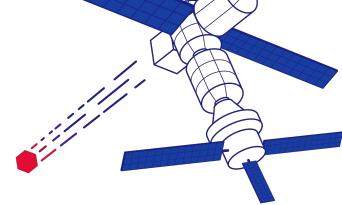




更快的查询响应

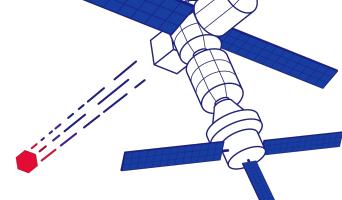
SELECT SUM(foo) FROM Table WHERE foo > bar



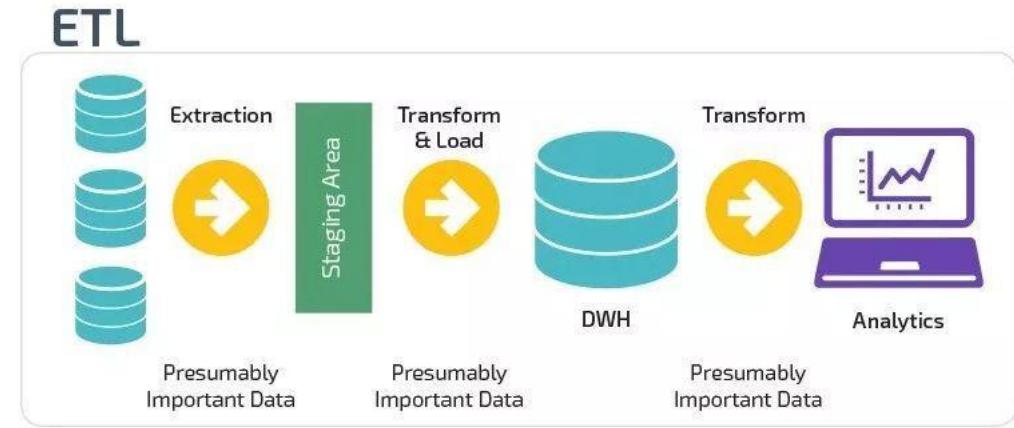
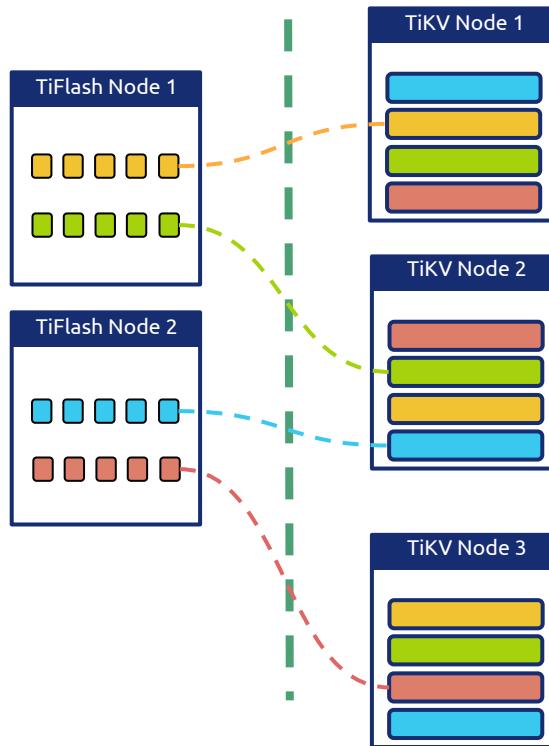


| 更快获取一致的最新数据

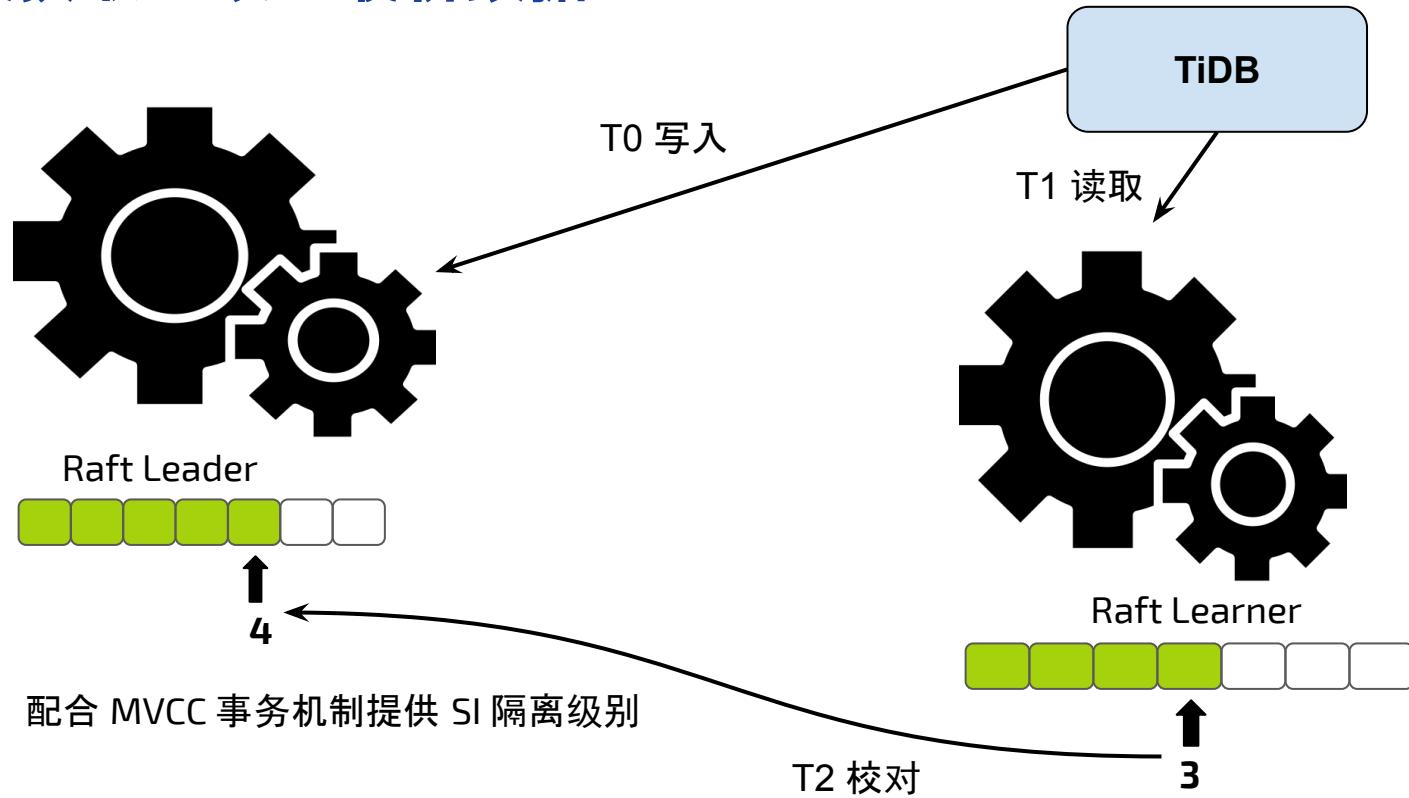
- 实时, 且自动容错 + 负载均衡的数据复制
 - 实时到账且对 TP 压力非常小
 - TiFlash 节点稳定性无关 TiKV
 - AP 查询对 TP 性能影响非常小
 - 自带负载均衡, 可多对多高效复制
 - 复制协议本身自带容错和数据恢复
 - 更快速的数据获取可以让业务决策更有效
 - 更快的物流周转, 更精确的风控, 更敏捷的商业策略调整

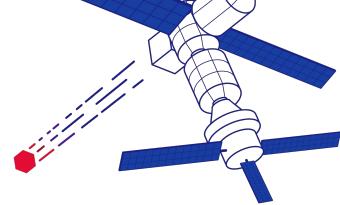


更快获取一致的最新数据



| 更快获取一致的最新数据

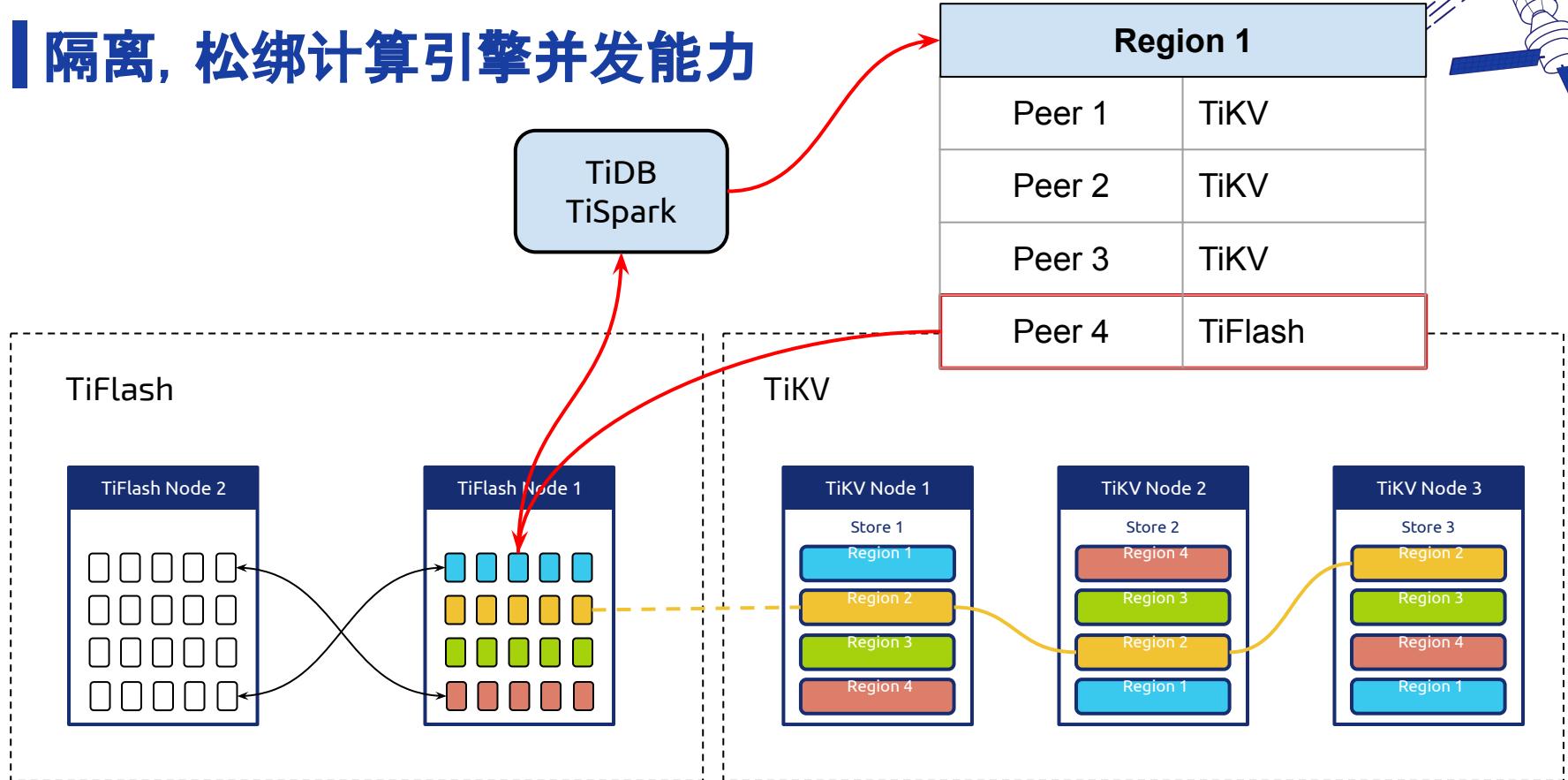


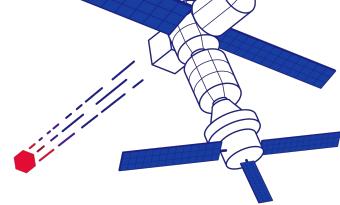


|| 隔离, 松绑计算引擎并发能力

- TiSpark 曾经是个麻烦
 - 火力全开, 调高并发 → TiKV 集群抖动乃至业务投诉
 - 佛系读取, 降低并发 → 集群稳定, 但查询速度极慢
 - 为 TiSpark 特别设计了降低读取效率的参数, 何其脑裂精分
- TiDB 直接查询在线数据也有类似问题
 - 为实时查询增加索引 → 避免大范围扫描造成抖动
 - 但却影响 TP 业务写入速度
 - 不增加索引吧 → 不但统计极慢, 且一旦做些统计业务延迟大增

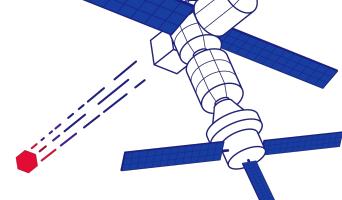
隔离, 松绑计算引擎并发能力





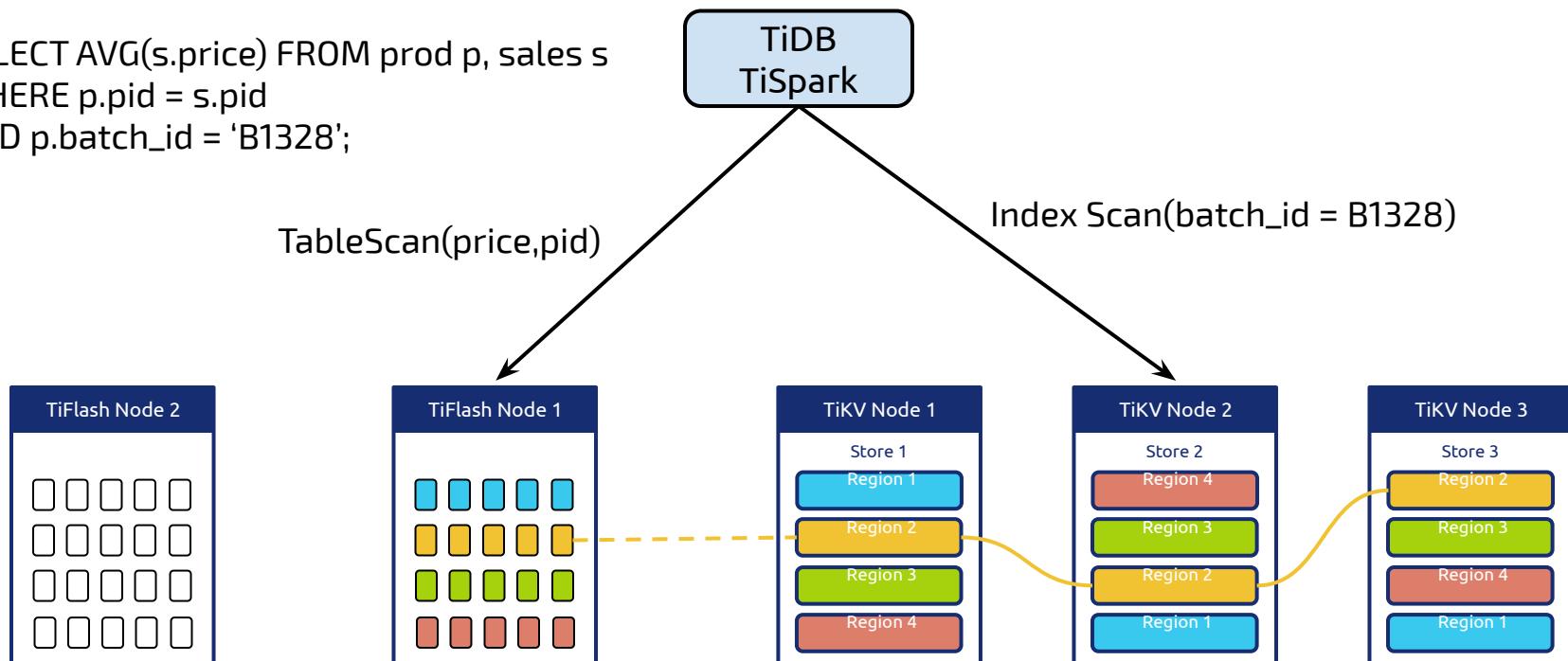
行列配合达到最优速度

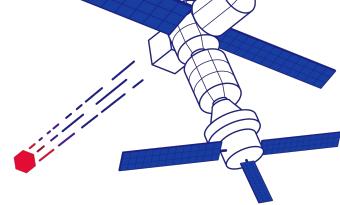
- 不开启隔离时, TiFlash 也可以作为列存索引使用
 - 作为一张表的特殊索引
 - 和行存及其现有索引机制联合加速
- 统一由优化器根据代价模型评估
 - 批量扫描且无行存索引或者索引无法有效 过滤数据时
 - 表过宽读取放大严重时



行列配合达到最优速度

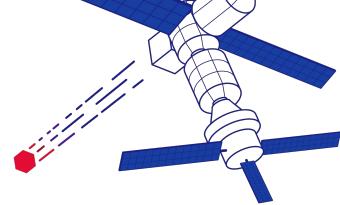
```
SELECT AVG(s.price) FROM prod p, sales s  
WHERE p.pid = s.pid  
AND p.batch_id = 'B1328';
```





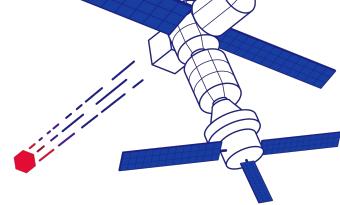
行列配合达到最优速度

- 行列混合加速, 而不仅仅是列存
 - 统一平台满足高并发短查询混合中低并发实时分析和跑批, 报表等业务
 - 甚至同一查询可同时使用行列进行加速达到最大效果
 - 例如物流数据平台
 - 大量高并发低延迟点查追踪包裹或者客户信息
 - 同时进行亚秒级的多维分析, 例如分省统计某种货物的发单率
 - 由于数据实时到账且一致, 两种用法数据完全不会打架
 - 同一套平台, 无需切换



| 更快的业务接入速度

- 亲身经历: 某游戏接入 Hadoop 平台进行分析
 - 3 天时间架设数据从分布式数据库传输变更日志的管道
 - 2 天时间编写将日志入库 Hive 的作业
 - 由于传输管道本身无法保证一致性, 作业本身需要去重, 且分区入库需要校对时间边界
 - 还需要编写数据校对代码, 以保证及时发现传输错误, 又过去几天
- TiFlash 让用户更早接入服务
 - 快到仅仅需要一条命令开启同步



| 更快的业务接入速度

```
mysql> ALTER TABLE orders SET TIFLASH REPLICA 2;
```