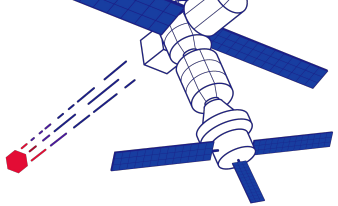
TiDB TechDay 2019

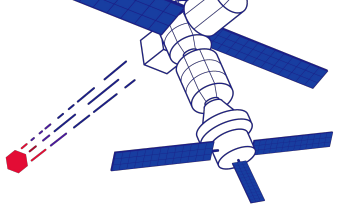
TiDB 新 AP 架构深度解析

TiDB AP 端的核心理念



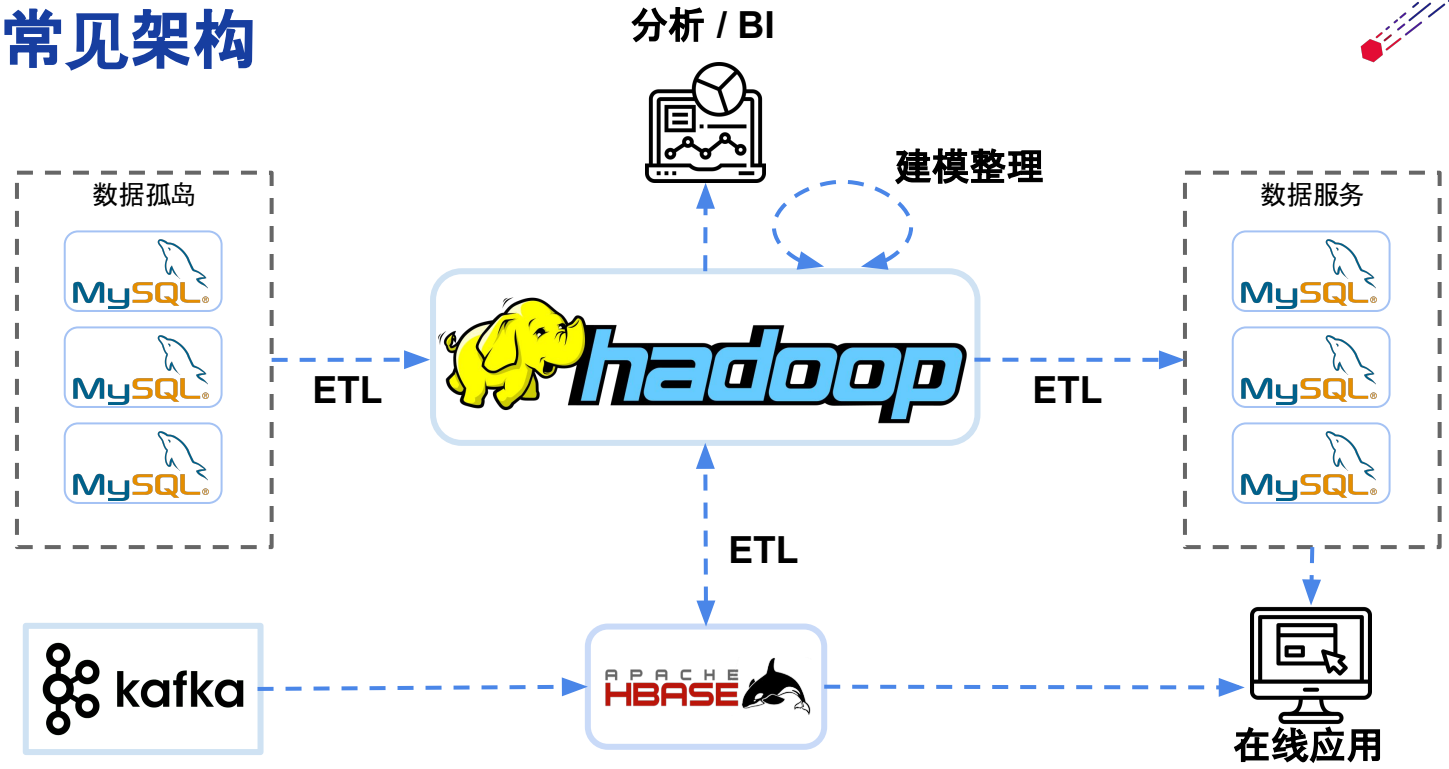
更简单，更敏捷

数据分析平台需求



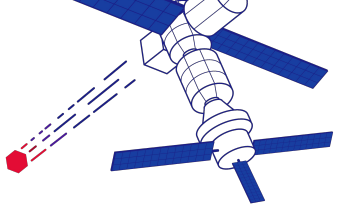
- 分析平台有诸多维度的需求
  - 可扩展的存储和计算力：数据汇聚
  - 中短程，中高并发查询：用于数据服务层，借助应用服务器以网页或移动设备提供展示
  - 中长程，中低并发查询：用于数据加工，模型构建等低频批量计算
  - 混合查询：即席查询，数据探索，或者综合查询业务；形式无法预估，尽可能快
  - 敏捷：尽可能简单，减少数据加工流转的周期，更实时的查询

一个常见架构



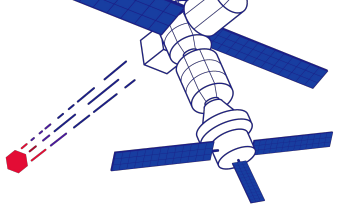
Icons made by Eucalyp, prettycons from Flaticon

为何如此复杂



- 各个组件特性不同，组合才能满足需求
  - 关系型数据：较小的存储规模；服务在线业务或者数仓订制处理后数据的高速高频展示
  - NoSQL：可扩展的大量数据存储，用于高速点查点写类在线应用
  - 数据湖 / 分析型数据库：可扩展的海量存储以及多样性的处理引擎，可以进行离线数据分析和报表计算，但很难直接服务于在线 / 高并发查询业务
  - 不同数据存储之间无法无缝衔接，需要数据移动
    - 更糟糕的是，类似 Hadoop 这样批量导入系统需要复杂的 ETL 才能同步变更数据
    - 接入一个新的数据源并不是简单的事情

我们希望它更简单

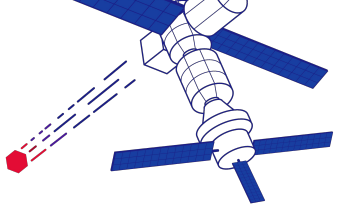


- TiDB 在 AP 端不断向简化架构演进
- 新组件和新功能（部分 WIP 将在 4.0 发布）
  - TiFlash
    - 作为隔离的列存引擎：提供媲美 Hadoop 的性能，以及对 TP 的隔离性
    - 作为整合的列存索引：提供 计算层更智能更高速的批量 读取能力
  - TiSpark 事务性写入：ETL 功能补充，Spark Streaming 接入
  - 全组件向量化加速：包含 TiKV，TiDB 以及新引擎 TiFlash

TiSpark 事务性写入

PingCAP TiDB

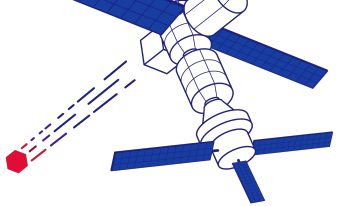
TiSpark 分布式事务性写入（WIP）



- TiSpark 的原生写入支持
  - 不再依赖 Spark 的 JDBC 数据源
  - 分布式写入和原子提交
  - 补充 TiDB 在 ETL 场景缺陷
  - 维持约束（索引，唯一性等）
- 简单易用
  - 兼容 Spark 原生风格
  - 自动类型转换
  - Region 预切割以负载均衡
  - 支持 Upsert（WIP）

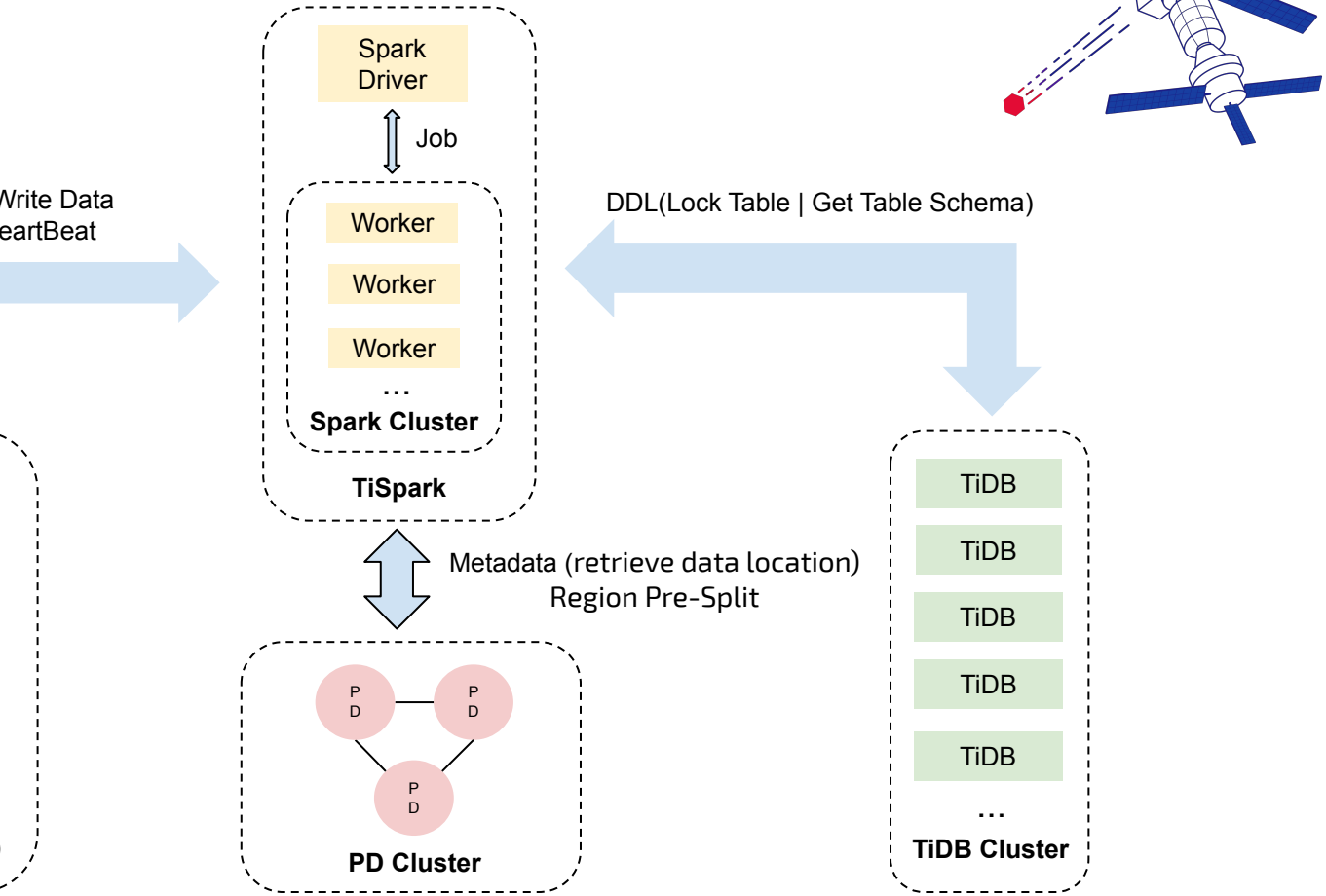
```
df.write
  .format("tidb")
  .option("database", "db")
  .option("table", "tbl")
  .mode("upsert")
  .save()
```

分布式写入 ACID 支持



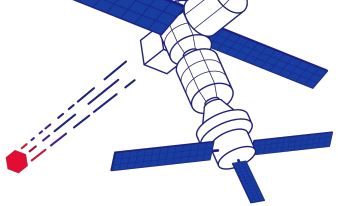
- 分布式二阶段提交
  - Driver 负责 Primary Row Prewrite 和 Commit
  - Executor 负责其他数据的 Prewrite 和 Commit
  - Secondary Commit 可选提交与否以加速 ETL 过程
- 原子性
  - 要么都成功要么全撤销
- 通过表锁维持事务规则

设计


Spark Driver
Job
Worker
Worker
Worker
...
Spark Cluster
TiSpark
Read | Write Data
TTL HeartBeat
DDL(Lock Table | Get Table Schema)
TiKV
TiKV
TiKV
TiKV
TiKV
TiKV
...
TiKV Cluster (Storage)
Metadata (retrieve data location)
Region Pre-Split
P D
P D
P D
PD Cluster
TiDB
TiDB
TiDB
TiDB
TiDB
...
TiDB Cluster

向量化

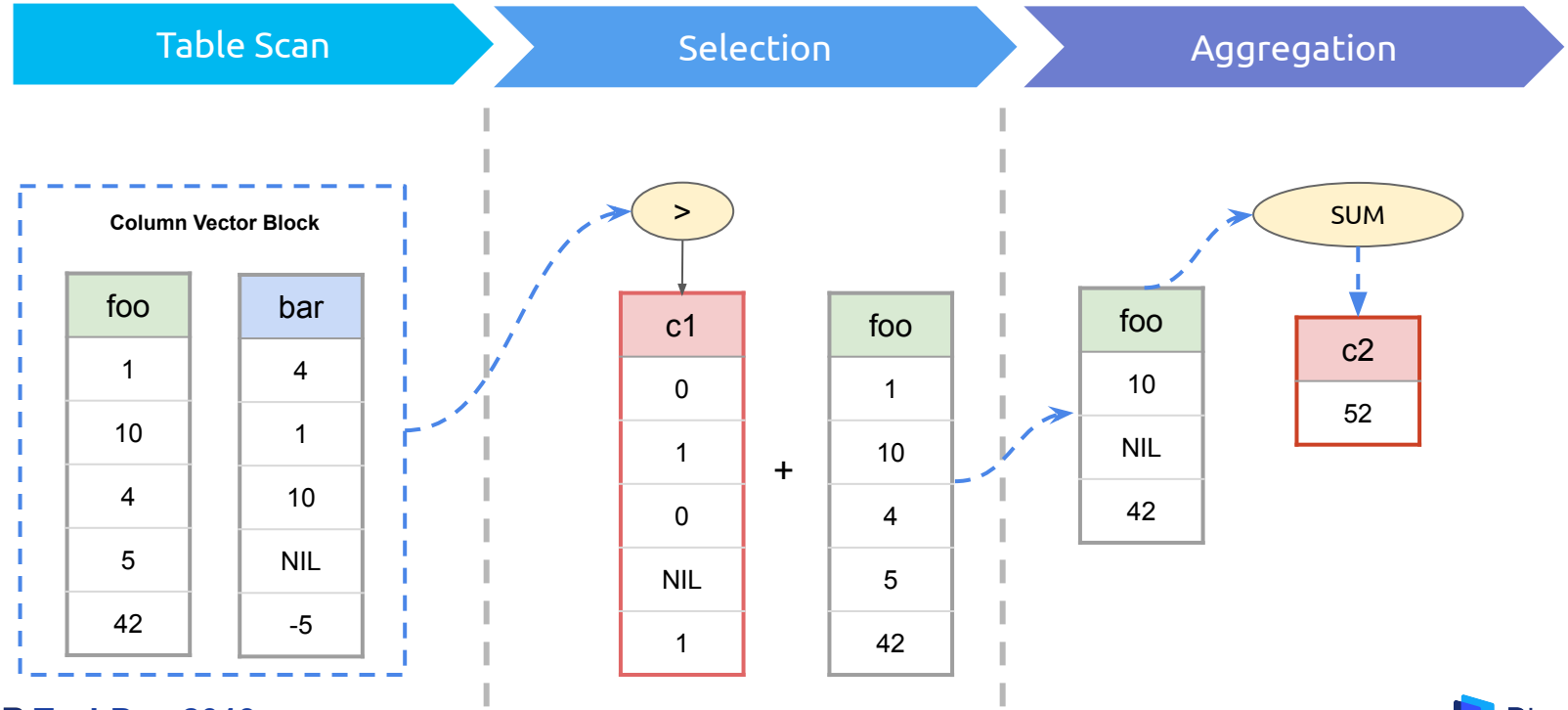
向量化支持



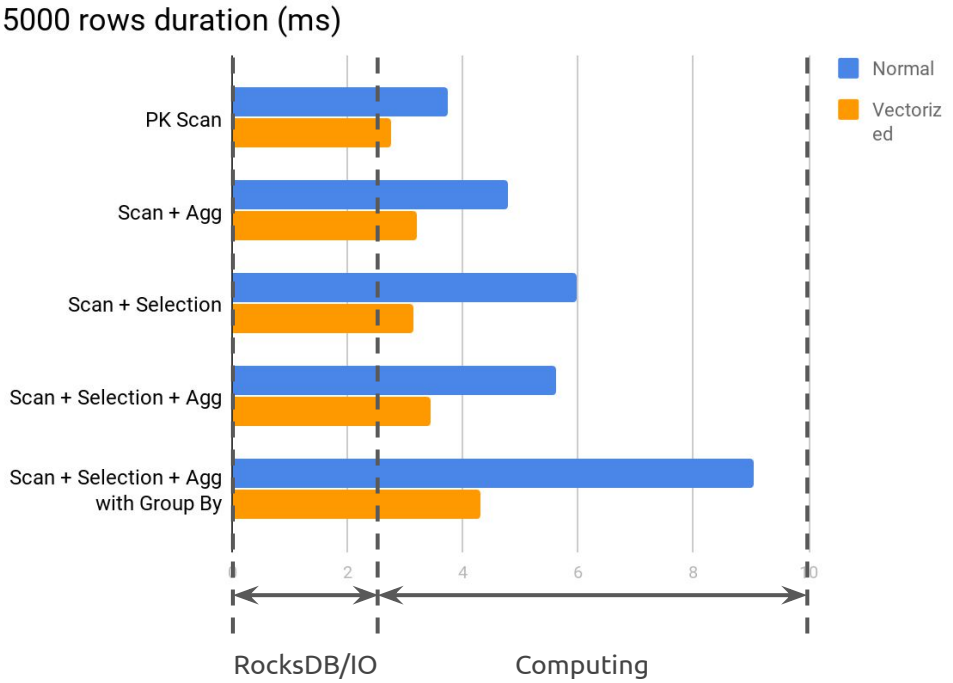
- 所有计算组件向量化支持
  - TiKV Coprocessor
  - TiFlash（借助 Clickhouse 原生实现）
  - TiDB（WIP，暂时是 Batch Execution，向量化中间状态）
- 大幅加强 TiDB 对大批量数据的计算效率
  - 加上之前已经实现的并发算子，大大提升 TiDB 在大量数据场景下的速度
  - 单机下超过 Spark 基于 Codegen 引擎的速度

向量化支持

SELECT SUM(foo) FROM Table WHERE foo > bar



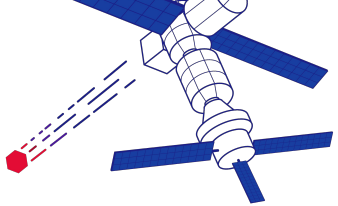
向量化支持 - TiKV Coproc



TiFlash 列存引擎

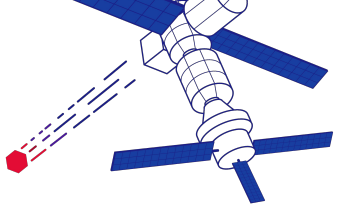
PingCAP TiDB

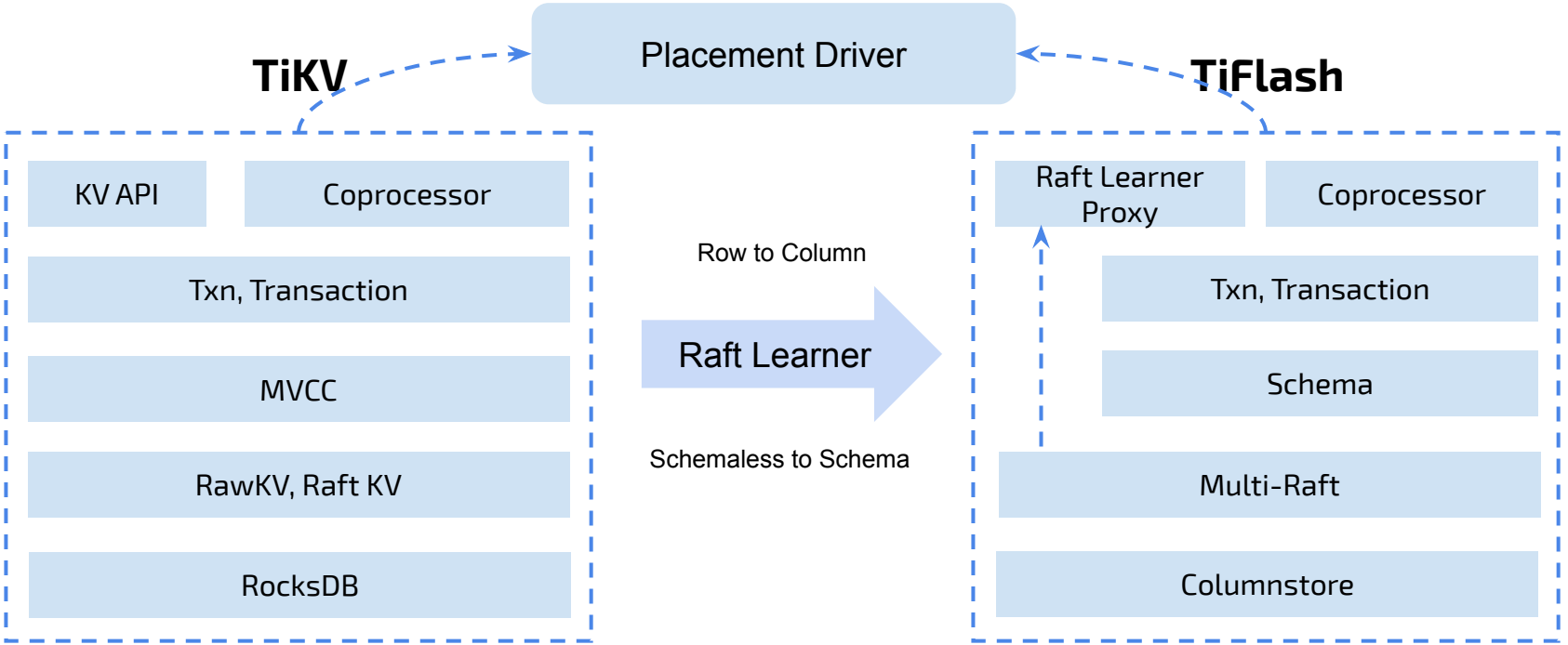
列存引擎 TiFlash



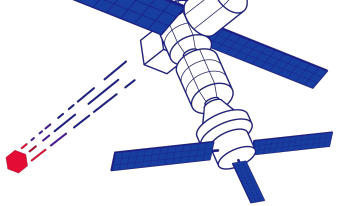
- 支持高频更新的列存引擎
- 以 Learner 角色接入 Multi-Raft 体系
  - 数据均衡
  - 强一致的读取
- 隔离性：作为隔离的 OLTP 数据列存镜像
- 整合性：作为主数据的列存索引（WIP）
- 部分基于 Clickhouse（感谢 Clickhouse 团队）
- 支持 HDD 存储

TiFlash 列存引擎



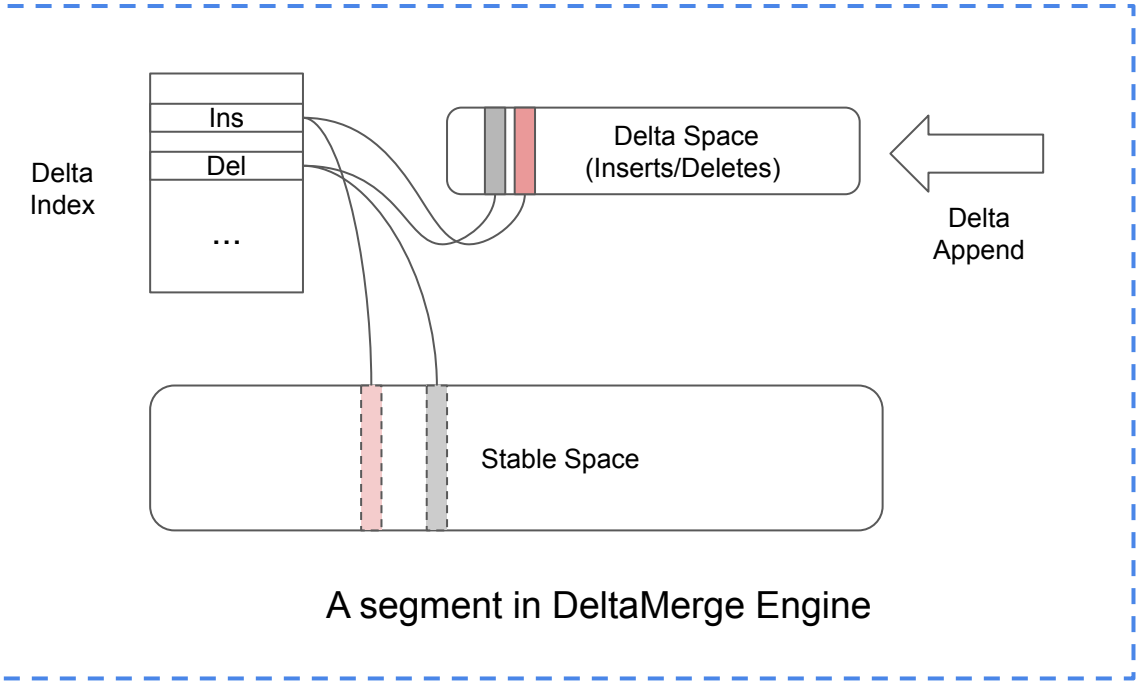


列存 + 高频更新 + MVCC

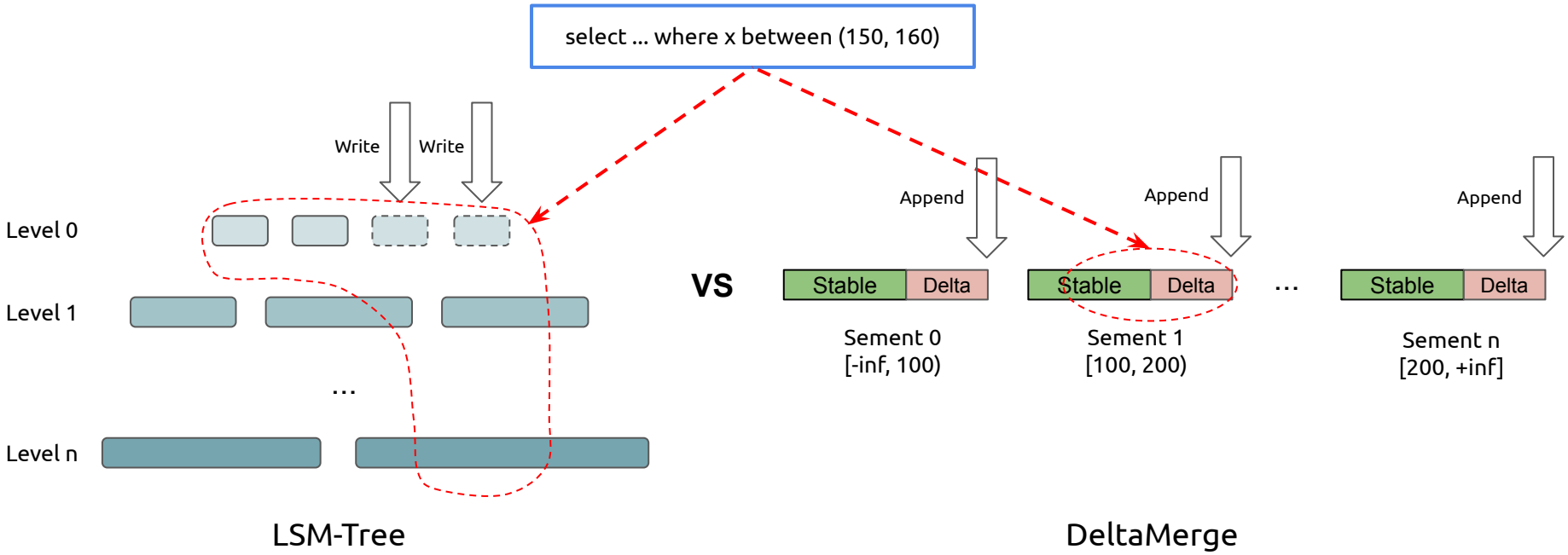


- 列存往往只支持低频批量更新
  - 一行拆多列使得传统更新方式性能非常差（Random IO 更多）
  - 列存压缩使得更新需要解压，改写，再写回去
- 我们还需要支持主键 Range Scan + MVCC 以对接计算层的统一读取
- GA 版将包含 LSM 列存正式版 + DeltaMerge 结构的 Beta 测试版
  - LSM 引擎基于 Clickhouse MergeTree 改造
  - DeltaMerge 引擎则是完全自研：相比 LSM 版降低写入以换取读速度

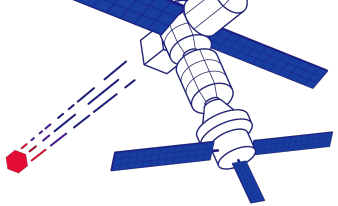
DeltaMerge 引擎



DeltaMerge vs LSM-Tree

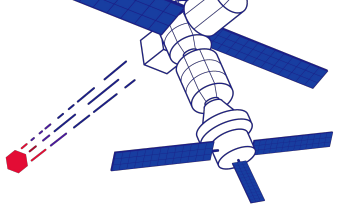


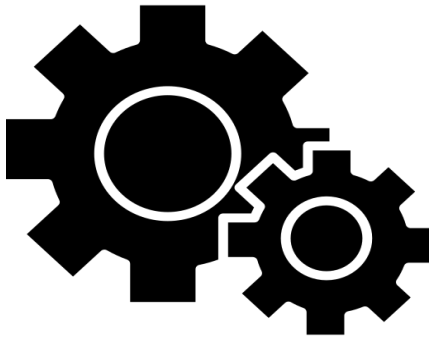
Raft Learner 协议



- 作为非投票的抄写角色加入 Raft 组
  - 保持异步同步，对 Leader 的压力非常小
- 通过 Region Raft Index 校验保持读时同步
  - 向 Leader 发起 Raft Log 进度校验，补完数据
  - Multi-Raft 下配合 MVCC 事务机制提供和 TiKV 一样的读取强一致性（SI 级别）
- 向额外的物理节点同步
  - 保持 AP 任务对 TP 服务的物理隔离

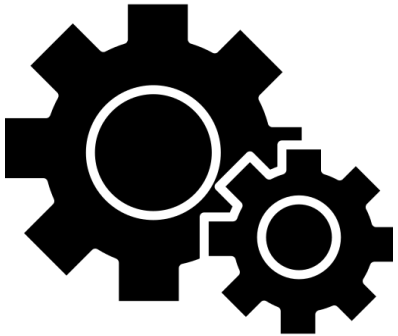
强一致读





Raft Leader

4



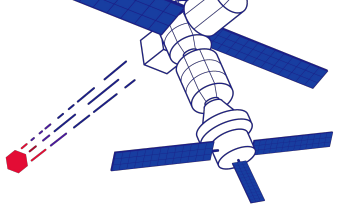
Raft Learner

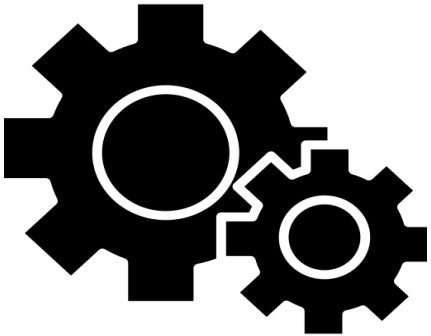
3

TiDB

虽然是异步写入，但是 读取时会进行 Raft Log 索引校验

强一致读

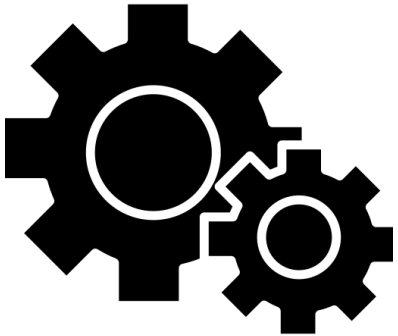




TiDB

Raft Leader

数据带有事务时间戳



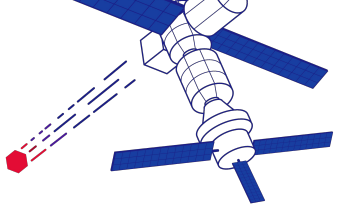
Raft Learner

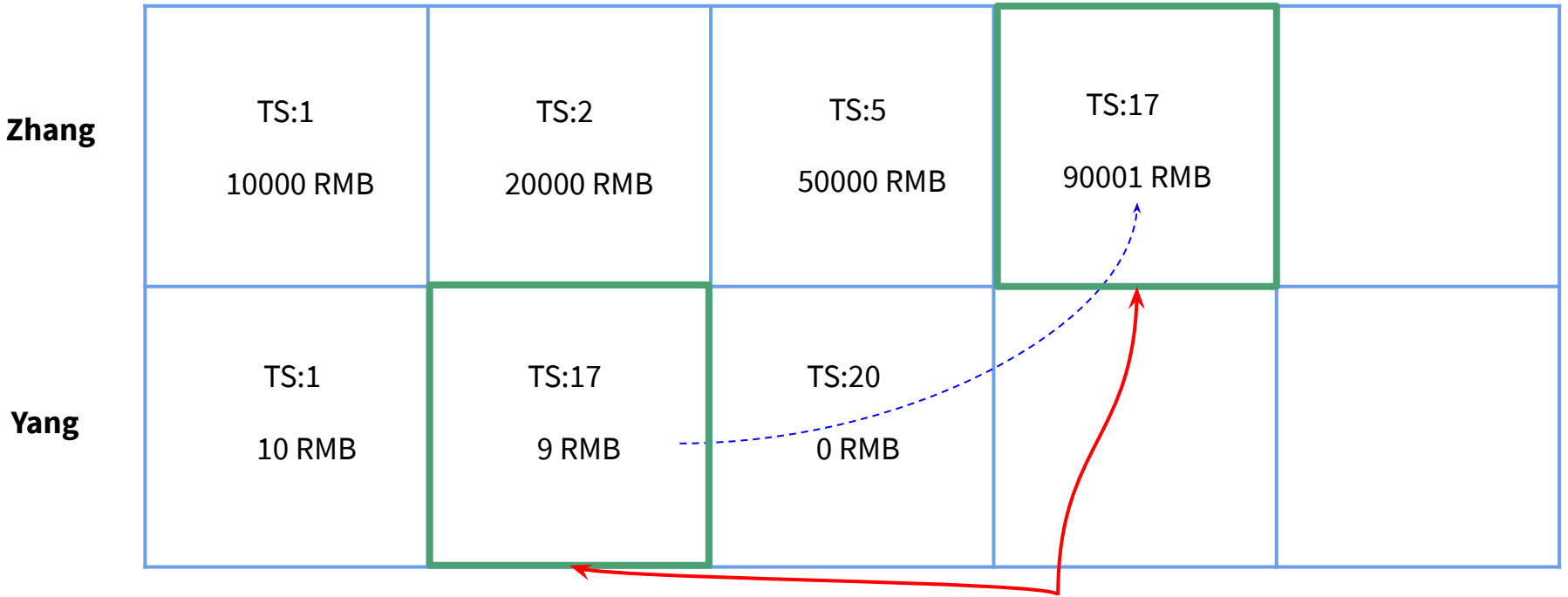
4

4

只有当 Raft Log 追上 Leader 的时候才真正提供 读取
配合 MVCC 机制提供强一致读取

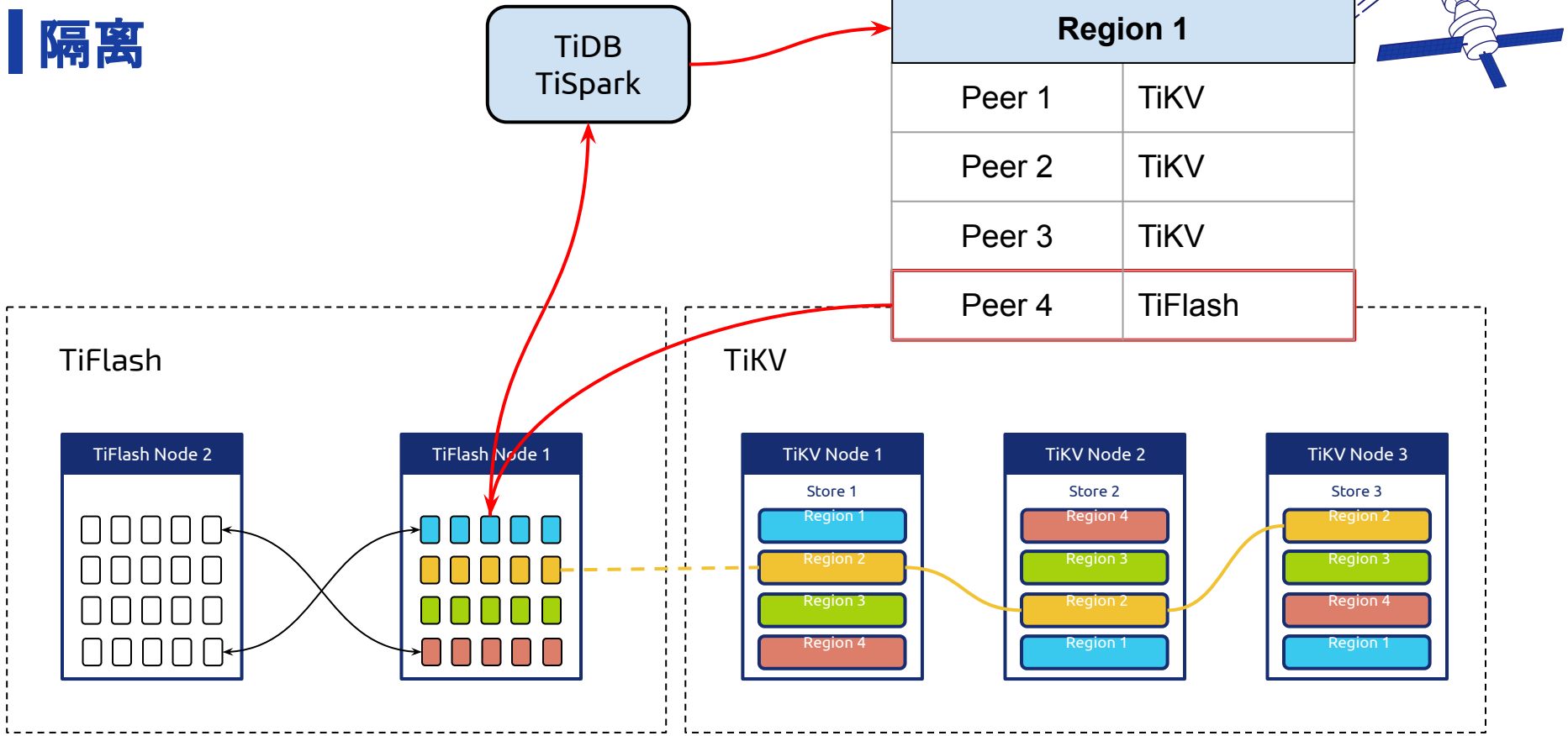
强一致读



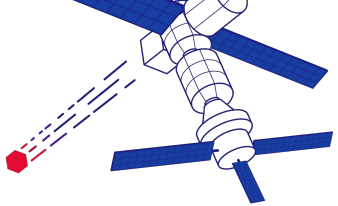


Timestamp = 17
Raft Index >= 4

隔离


TiDB
TiSpark
Region 1
Peer 1
TiKV
Peer 2
TiKV
Peer 3
TiKV
Peer 4
TiFlash
TiFlash
TiFlash Node 2
TiFlash Node 1
TiKV
TiKV Node 1
Store 1
Region 1
Region 2
Region 3
Region 4
TiKV Node 2
Store 2
Region 4
Region 3
Region 2
Region 1
TiKV Node 3
Store 3
Region 2
Region 3
Region 4
Region 1

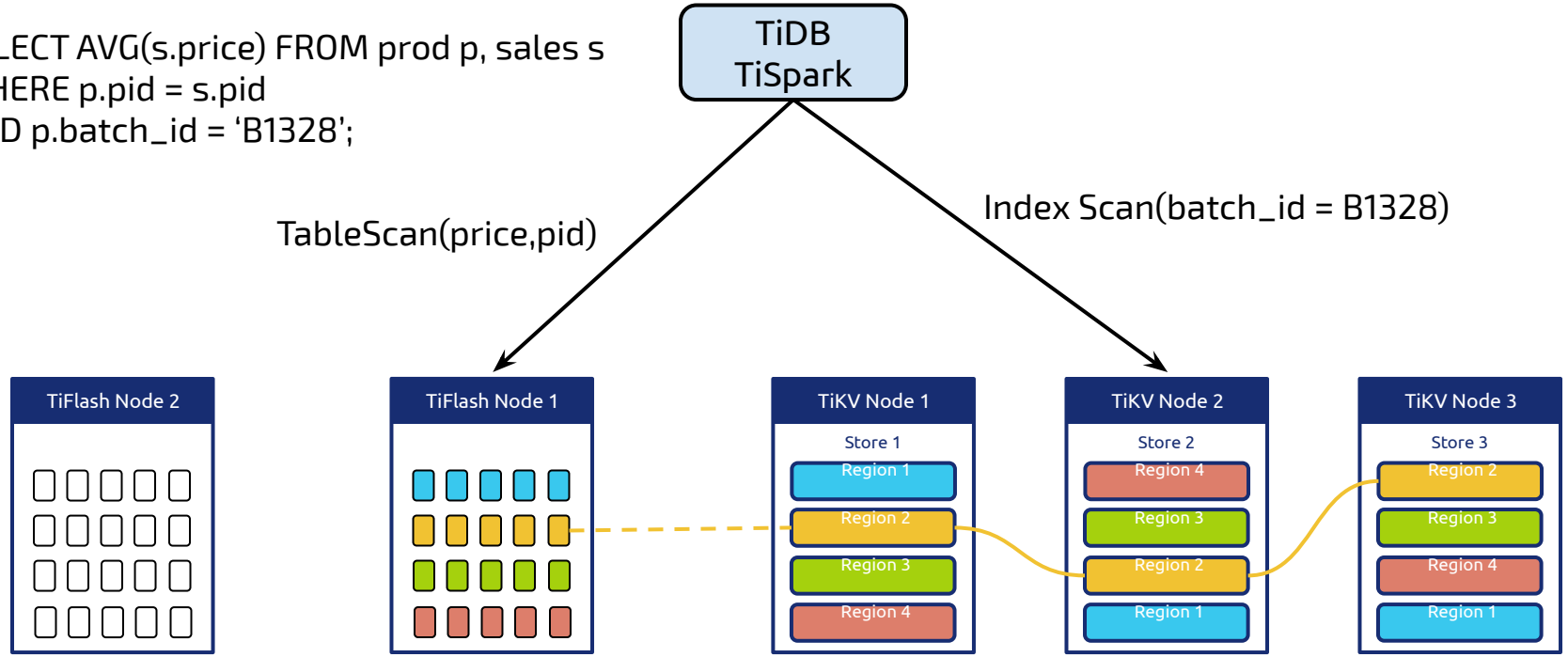
列存索引（WIP）



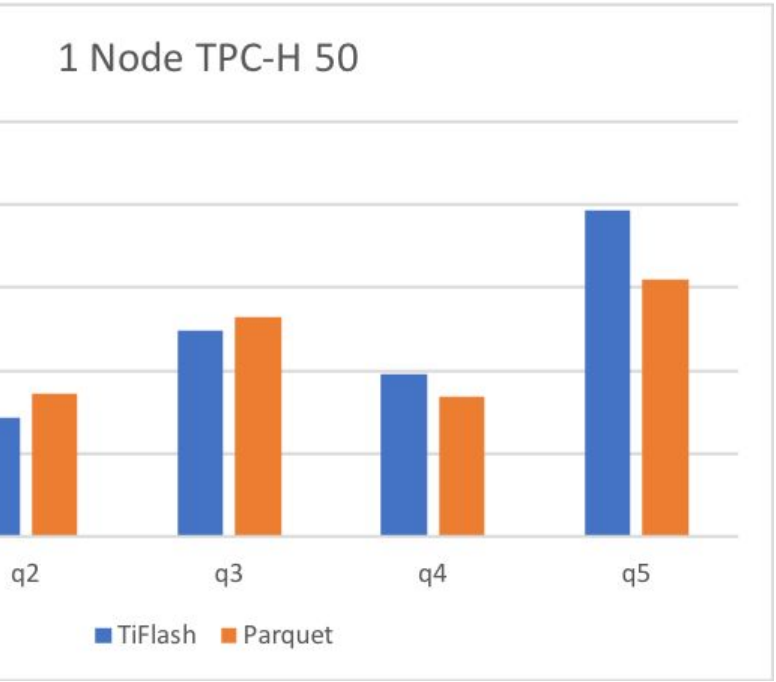
- 纯 AP 场景下，TiFlash 也可以作为列存索引使用
  - 作为一张表的特殊索引
  - 和行存及其现有索引机制联合加速
- 统一由优化器根据代价模型评估
  - 批量扫描且无行存索引或者索引无法有效 过滤数据时
  - 表过宽读取放大严重时

行列联合查询

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



性能（LSM 版）

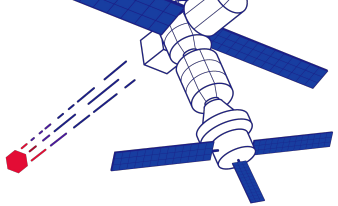


TiFlash Demo

PingCAP TiDB

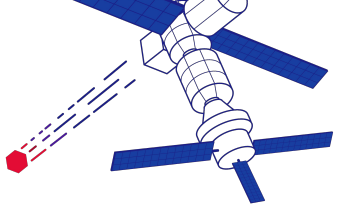
场景

混合负载分析平台



- 中台数据实时汇聚
  - TiDB 的传统 AP 场景，大规模实时 OLTP 数据同步
  - 简便的 MySQL 数据接入
  - 通过索引和协处理器支持高效的中短程高并 发查询
- 使用 TiFlash
  - 即席查询和复杂分析查询：无法预测，无法通过索引有效过滤
  - 联合行存索引提供 优于 Hadoop 的相应速度
  - 配合 TiSpark 进行数据加工并回写

HTAP

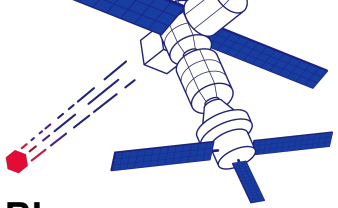


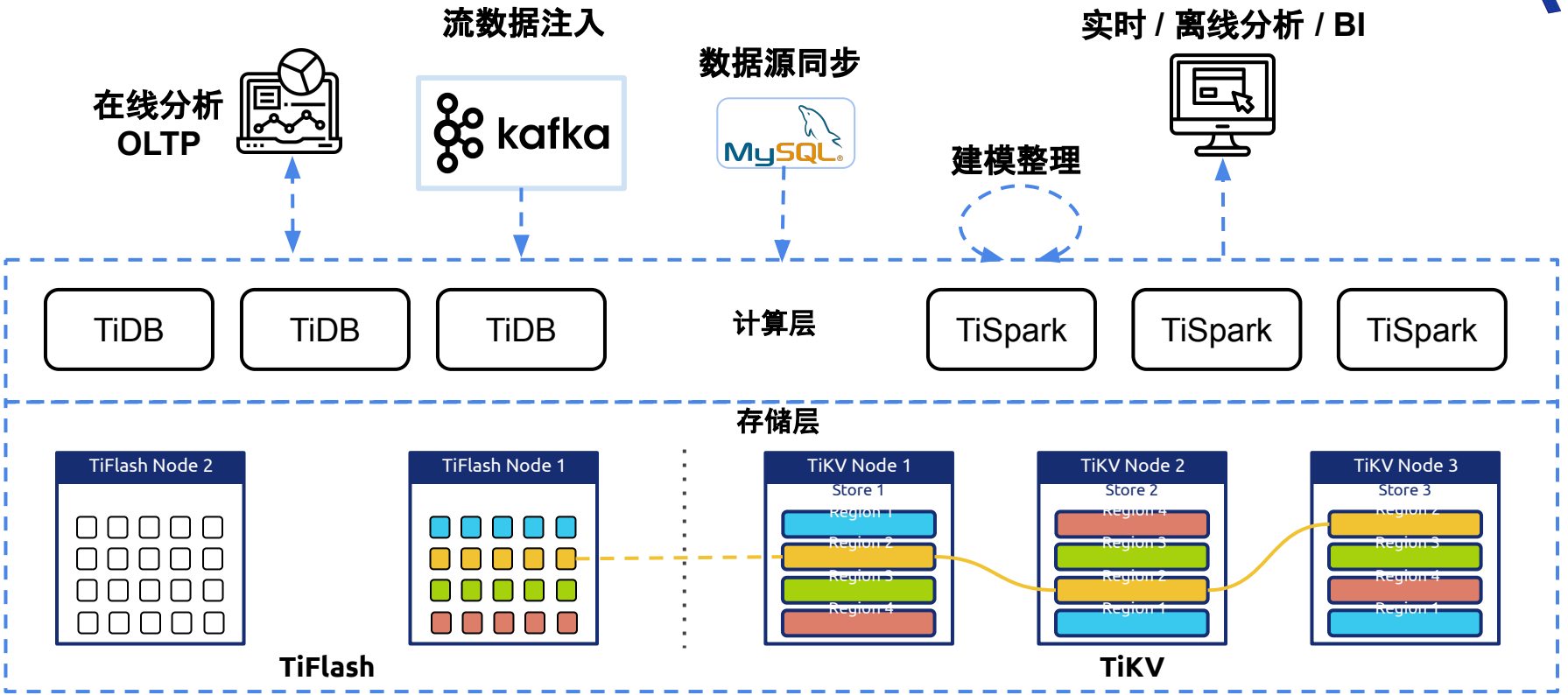
- 借助 TiFlash 进行 HTAP 实时查询分析
  - 物理隔离对 OLTP 几乎无影响
  - 线性一致性和快照隔离支持一致性要求高的 实时分析场景
  - 完全无需数据搬运，在最新 鲜的数据上进行分析
- 交易数据实时分析
  - 交易对账
  - 实时风控

野心

PingCAP TiDB

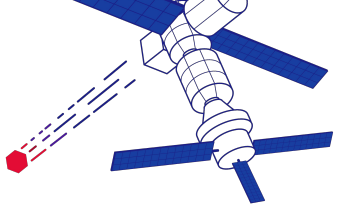
统一架构





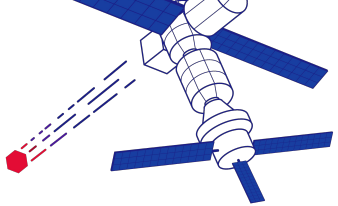
Icons made by Eucalyp, prettycons from Flaticon

优势



- 整个架构允许海量数据存储，可以实时汇聚多数据源
- 支持数据修改，可以进行数据临时修正
- TiDB 可以直接处理 OLTP 业务，可以更进一步缩短数据链路
- TiDB 配合 TiKV 利用行存加索引进行过滤，支持高并发低延迟查询
- TiDB / TiSpark 配合 TiFlash 列存进行无索引即席查询以及报表业务
    - 架构提供线性一致快照隔离的查询保障
    - 数据实时到账，进一步增加实时性和便捷性
- 优化器允许行列混合查询应对不可预测的用户需求

到 2019 年底我们还将缺少什么



- MPP 引擎
  - 让计算层更快更少 OOM，让 TiDB 成为大数据量查询的主要入口
  - 比 TiSpark 更适合非 ETL 类复杂 SQL 查询的引擎
- TiFlash 非同步表写入
  - 让用户可以直接多副本写入 TiFlash 列存
  - 让数仓场景能很大程度不再依赖 Hadoop
- 统一计算层的权限系统

Thank You !

Icons made by Eucalyp, prettycons from Flaticon