
Architect
SACC
2022 中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2022
激发架构性能 点亮业务活力
云上会议 网络直播 | 2022年10月27-29日
ITPUB

开源机器学习数据库OpenMLDB：线上线下一致的高可用特征平台

张 浩
第四范式 资深系统架构科学家
OpenMLDB PMC

About Me

张 浩

zhanghao@4paradigm.com

- 开源项目 OpenMLDB PMC
- 第四范式资深系统架构师
- 博士毕业于新加坡国立大学计算机系

目录

1. 人工智能工程化落地的数据和特征挑战

正确、高效的 AI 数据和特征供给成为数据侧的新挑战

95%

时间精力花费在数据上

Source: How to Operationalize Machine Learning and Data Science Projects, Gartner

≠ AI

实时机器学习决策，需要毫秒级的数据和特征计算能力

两大 AI 应用：感知类、决策类

硬实时计算真正满足决策需求 – 实时数据、实时计算

流式计算为 Big Data 和 BI 设计

分钟/小时 级别 | Seconds 秒级 | Milliseconds 毫秒级

批量计算 Batch → 流式计算 Streaming → 硬实时计算 Hard Real Time

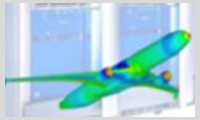
python, Spark Streaming, Flink, R, sas, APACHE STORM, APACHE kafka, TensorFlow

现在市面上所谓的AI实时计算大都是流式计算


AI无人车

AI事中反欺诈


量化交易


航空航天

硬实时场景蕴藏巨大商业价值，鲜有通用商业化产品

银行要求毫秒级业务响应

以某银行反欺诈场景为例

客户需求：

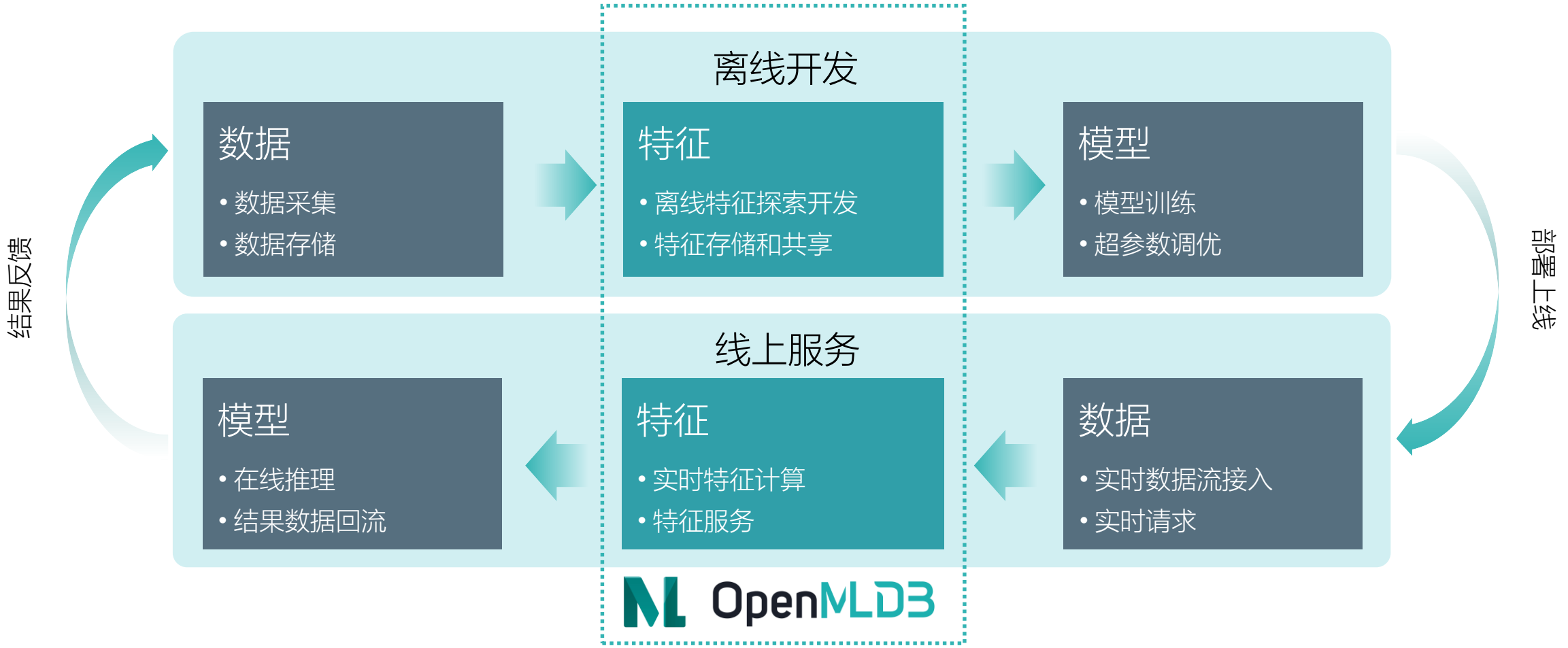
响应时间 20ms 内、高准召率的事中　反欺诈系统

解决方案	响应时间	准召率
传统规则系统	~200ms	较差
客户自研系统	~50ms	中等
第四范式先知	<20ms	优等

产品支撑：

高性能、高准确率的实时特征数据系统

机器学习应用从离线开发到上线全流程



实时事中反欺诈交易的特征计算

卡号	刷卡金额	刷卡时间
012159	1000	2022/01/12 08:00:00

历史交易表

卡号	刷卡金额	刷卡时间（已排序）
012112	223	2022/01/12 02:00:00
012159	15	2022/01/12 06:00:00
012159	1000	2022/01/12 07:59:55
012159	2000	2022/01/12 07:59:57
012159	1000	2022/01/12 08:00:00

计算模式
基于窗口聚合

3h
10s

特征计算

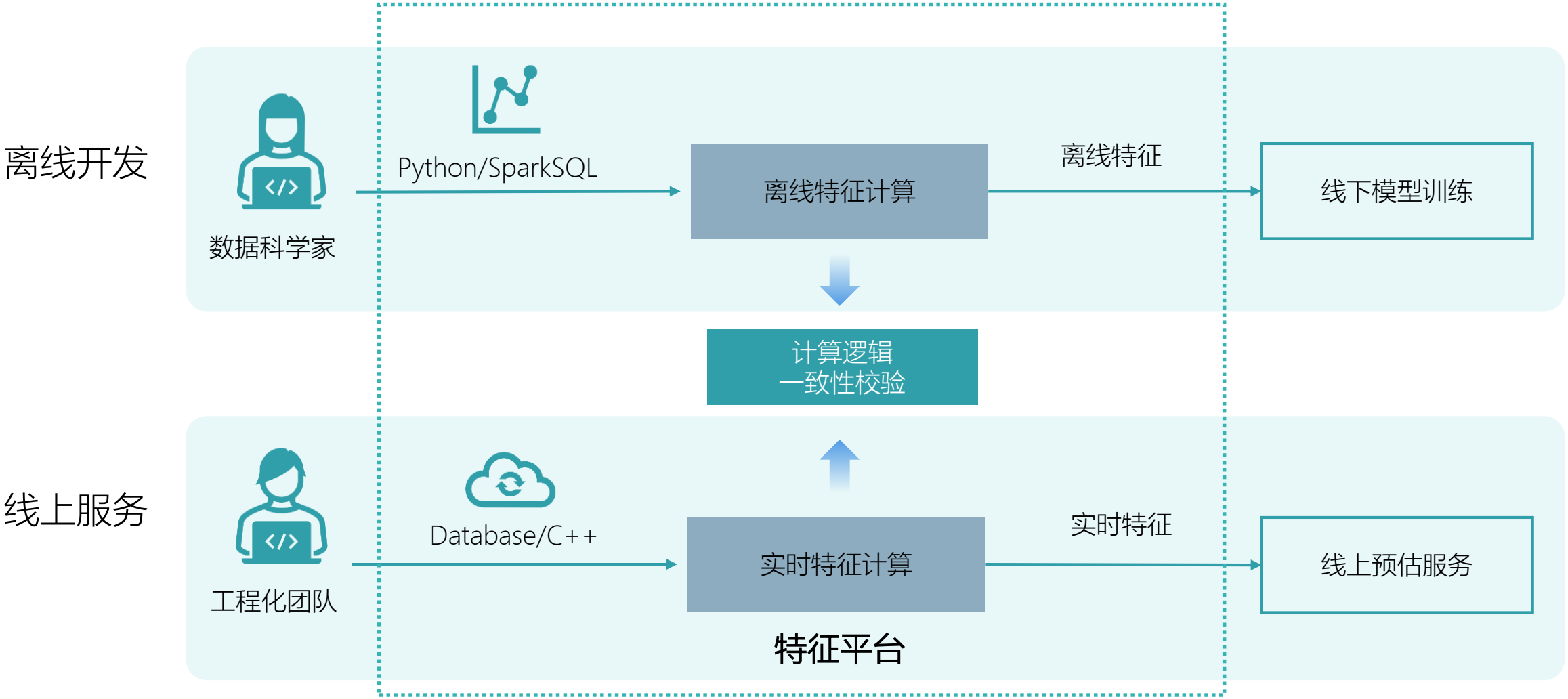
工程化需求

1. 线上线下一致性
2. 低延迟、高并发、高可用

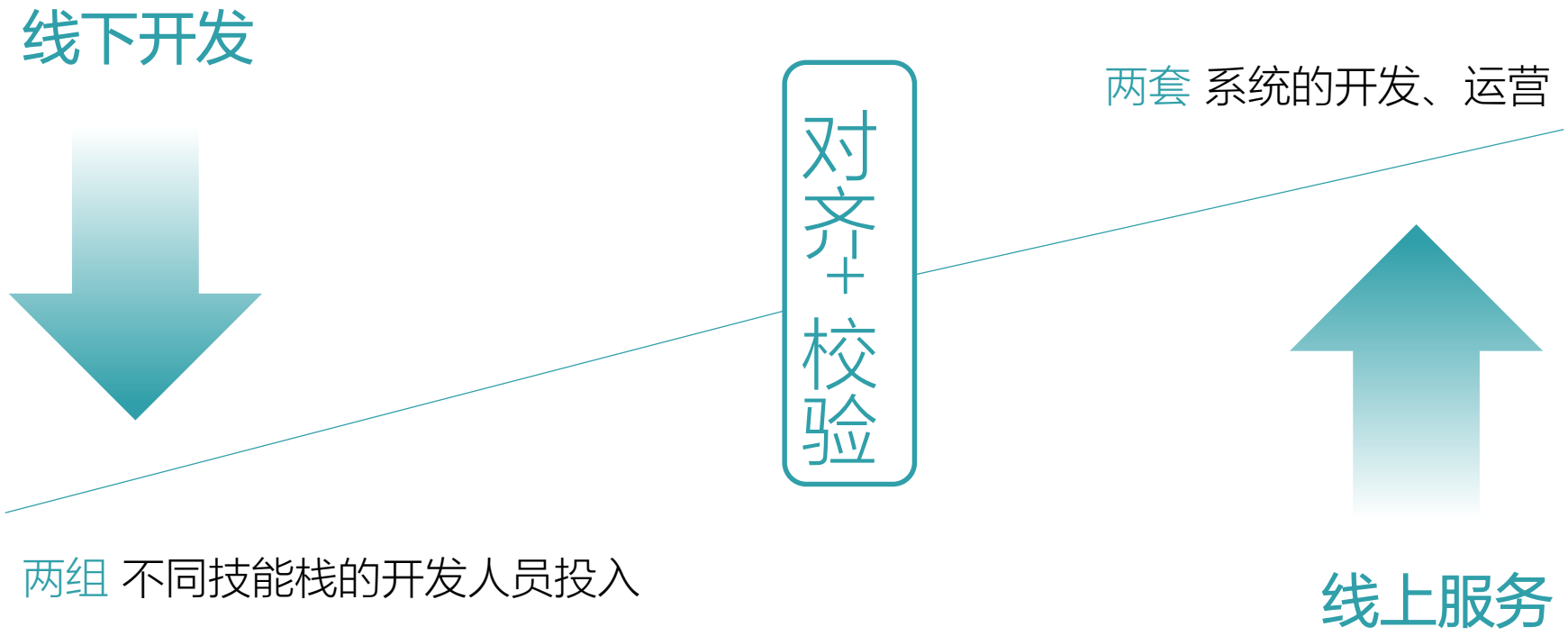
卡号	刷卡金额	过去10秒内：刷卡次数/刷卡最大金额/最小金额/平均金额	过去三小时内：刷卡次数/刷卡最大金额/最小金额/平均金额
012159	1000	3 | 2000 | 1000 | 1333	4 | 2000 | 14 | 1003

模型推理

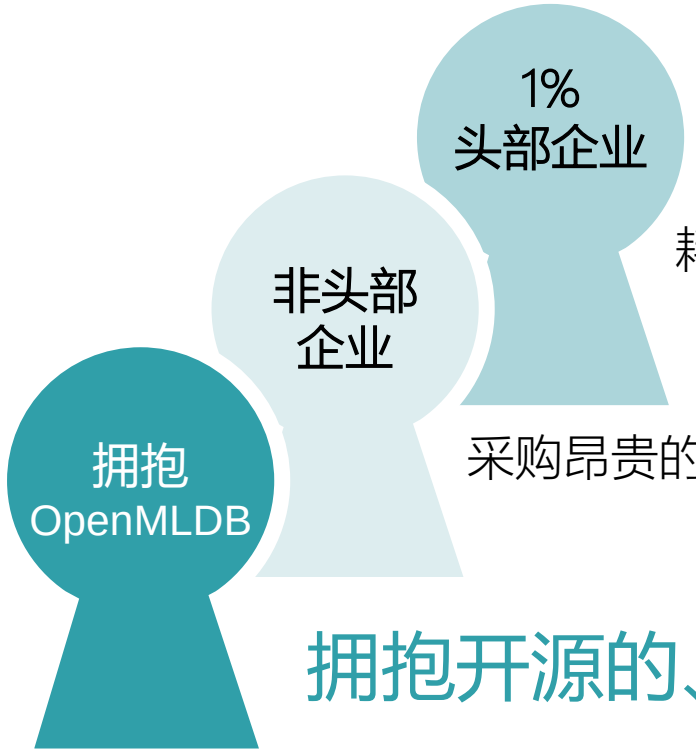
特征计算开发到上线全生命周期



线上线下一致性校验带来的高昂工程化落地成本



特征平台工程化解决方案



耗费上千小时自研构建平台

采购昂贵的 SaaS 工具和服务

拥抱开源的、低成本、高效的企业级解决方案

2. OpenMLDB：线上线下一致的生产级特征平台

过往5年 RTIDB/FEDB → OpenMLDB

在 100+场景 落地，覆盖超过 300个节点

信用卡现金分期精准营销	贷前风险评分	营销获客	个性化推荐	反洗钱可疑交易智能识别
信用卡账户风险预警	合规额度决策	风险管理	投顾客户挖掘	信用卡申请反欺诈
欺诈养卡防控	理财个性化推荐	零售贷款反欺诈	历史客户激活	客户流失预警
网点流量预测	交易欺诈评分	现金分期个性化推荐	信用卡交易反欺诈	金融产品推荐

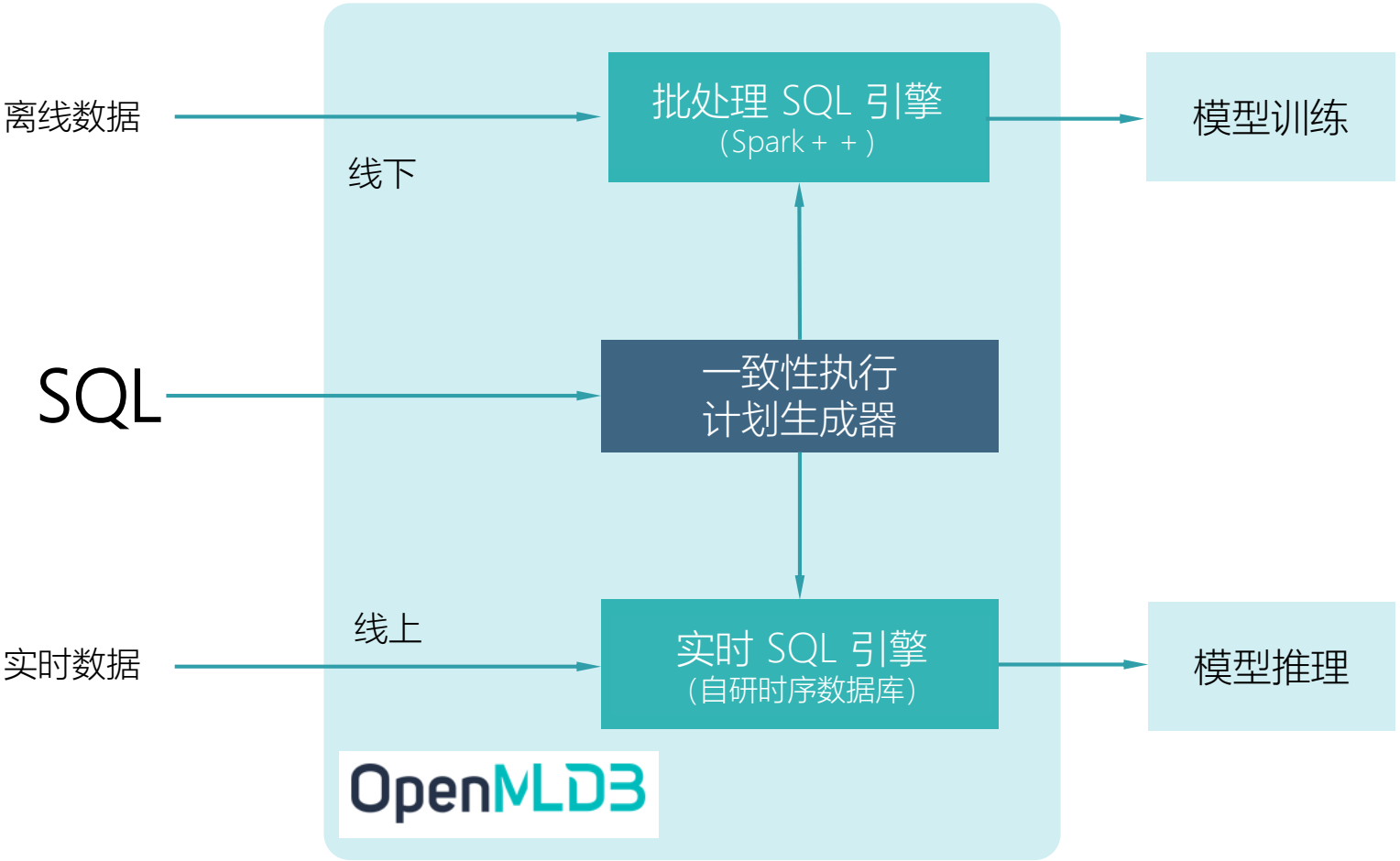
OpenMLDB：开源机器学习数据库，提供线上线下一致的特征平台

开发即上线

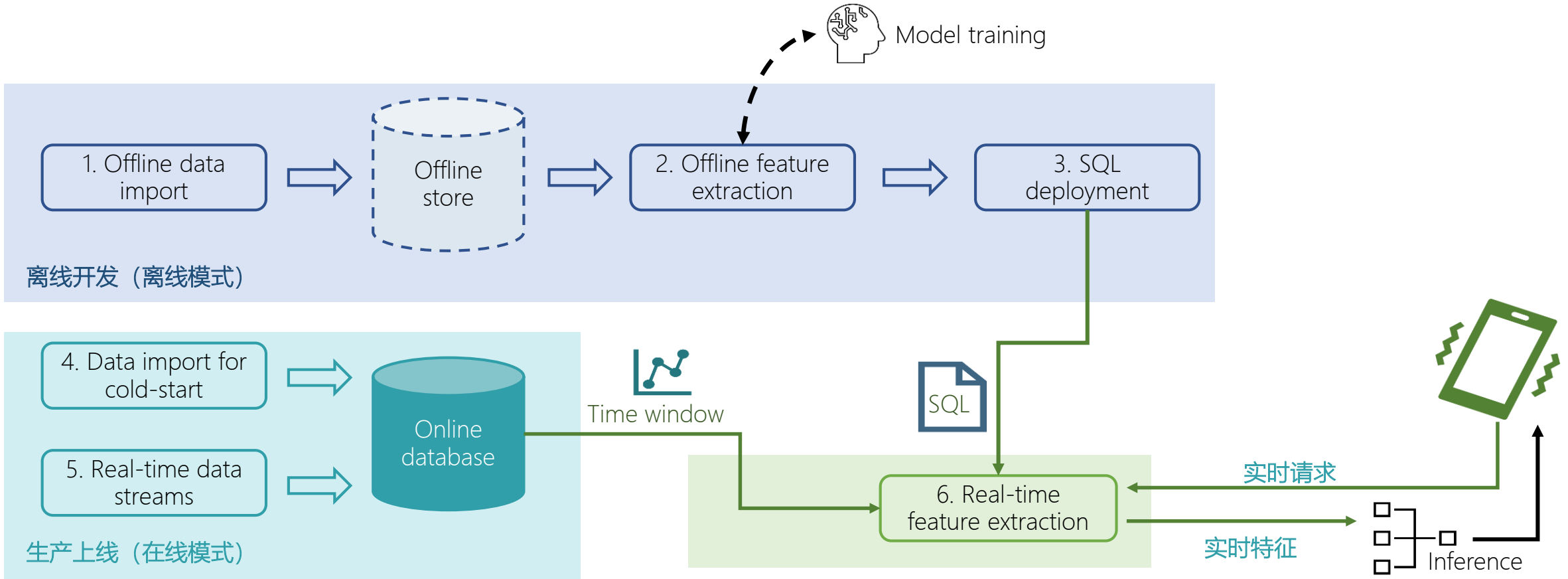
离线特征开发

一键上线

接入实时数据



从离线开发到线上服务完整流程

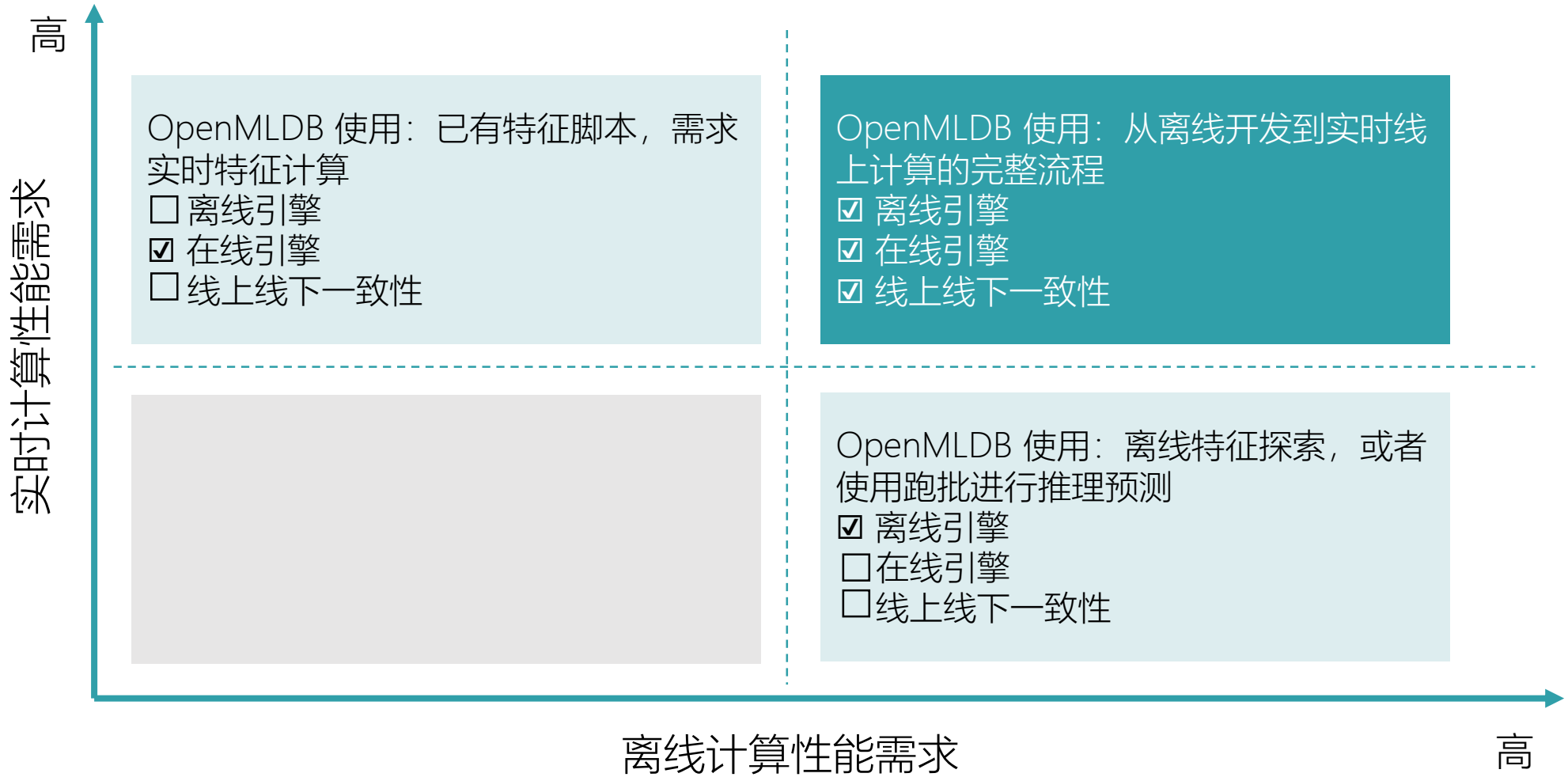


解决一个核心问题，提供一个核心特性

核心问题

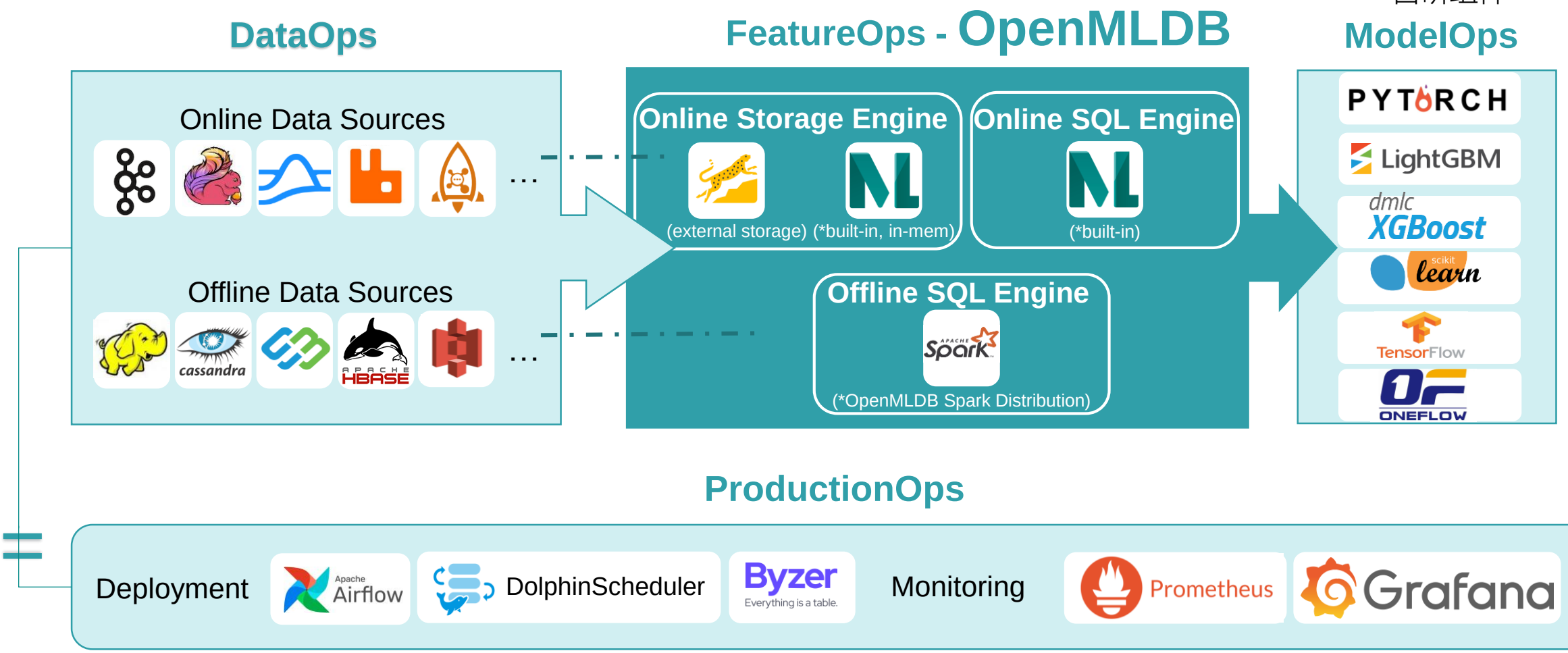
核心特性

OpenMLDB 应用场景和使用方式

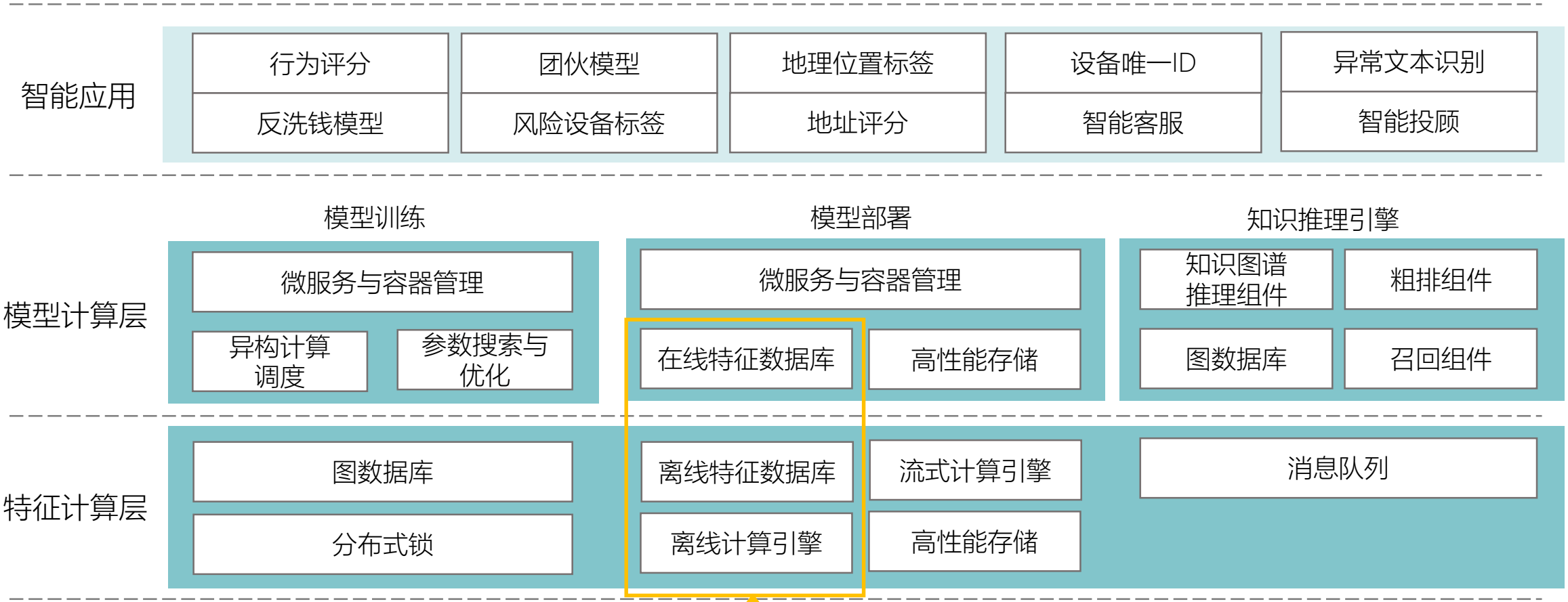


OpenMLDB 开源生态

* 自研组件



OpenMLDB 案例 – Akulaku 智能计算架构中的特征平台



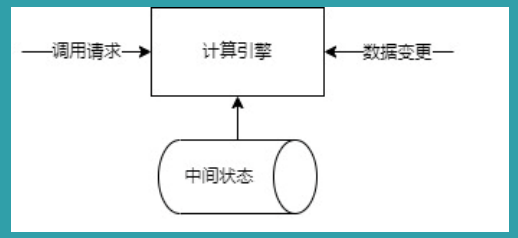
OpenMLDB 对 10 亿条订单进行窗口特征计算，达到 4 毫秒延迟性能

特征计算环节难点

- 线上部署：低延迟，高时效性，尽可能反映实时数据变更
- 线下分析：高吞吐量
- 逻辑一致：线下分析和线上部署的逻辑需要完全一致

OpenMLDB 解决方案

- **场景驱动**：业务调用环节驱动，实时计算结果，现用现算
- 具体方案：1）使用 SQL 作为离线和在线的桥梁
  2）在线基于时序数据库做时间滑窗




```
(product_id:1,  price:2    ,update_time:100L )
(product_id:1,  price:3    ,update_time:200L )

(product_id:2,  price:3    ,update_time:1000L )
(product_id:2,  price:4    ,update_time:1201L )
(product_id:2,  price:3.5  ,update_time:1100L )
```

```
SELECT COUNT(*)
FROM w100ms

WINDOW w100ms AS
 (PARTITION BY product_id ORDER BY update_time
 ROWS_RANGE BETWEEN 100ms PRECEDING
AND
CURRENT ROW)
```

基于 OpenMLDB 的业务实现

- 场景：近1天订单个数实时计算
- 数据量：10亿条订单数据/天
- 需求：实时更新，时间窗口实时滑动，存在复杂关联需求
- 测试结果：4毫秒 延迟

OpenMLDB 在 37 手游用户流失预测场景的应用

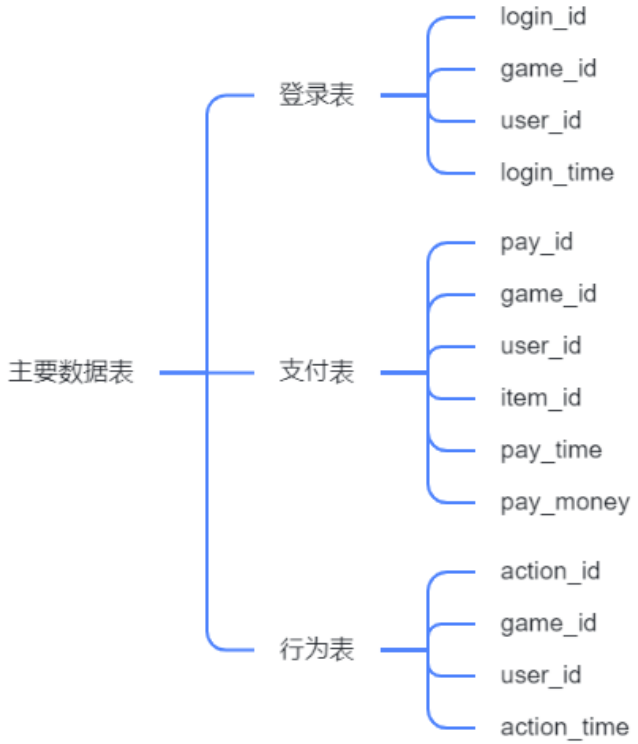
业务场景

什么时候	用户在触发登录的时候
做什么	预测用户是否会在最近7天内流失
用户量级	百万日活

组件部署

OpenMLDB 在 37 手游用户流失预测场景的应用

SQL 设计



特征设计

- 最近3，7，15，30天登录次数
  -- 主表窗口特征...
- 最近3，7，15，30天支付次数、支付金额
  -- 支付副表聚合特征...
- 最近3，7，15，30天行为次数
  -- 行为副表聚合特征...
- 这里展示的是简化后的特征，主要是展现部署流程

实验结论

- **离线导出**
  - 快速对特征进行计算和处理
  - 不需要担心数据泄漏等问题
- **在线服务**
  - 一键部署，减少传统流程中关于新特征上线的开发工作量
  - 上线周期从15/人/日->1.5/人/日

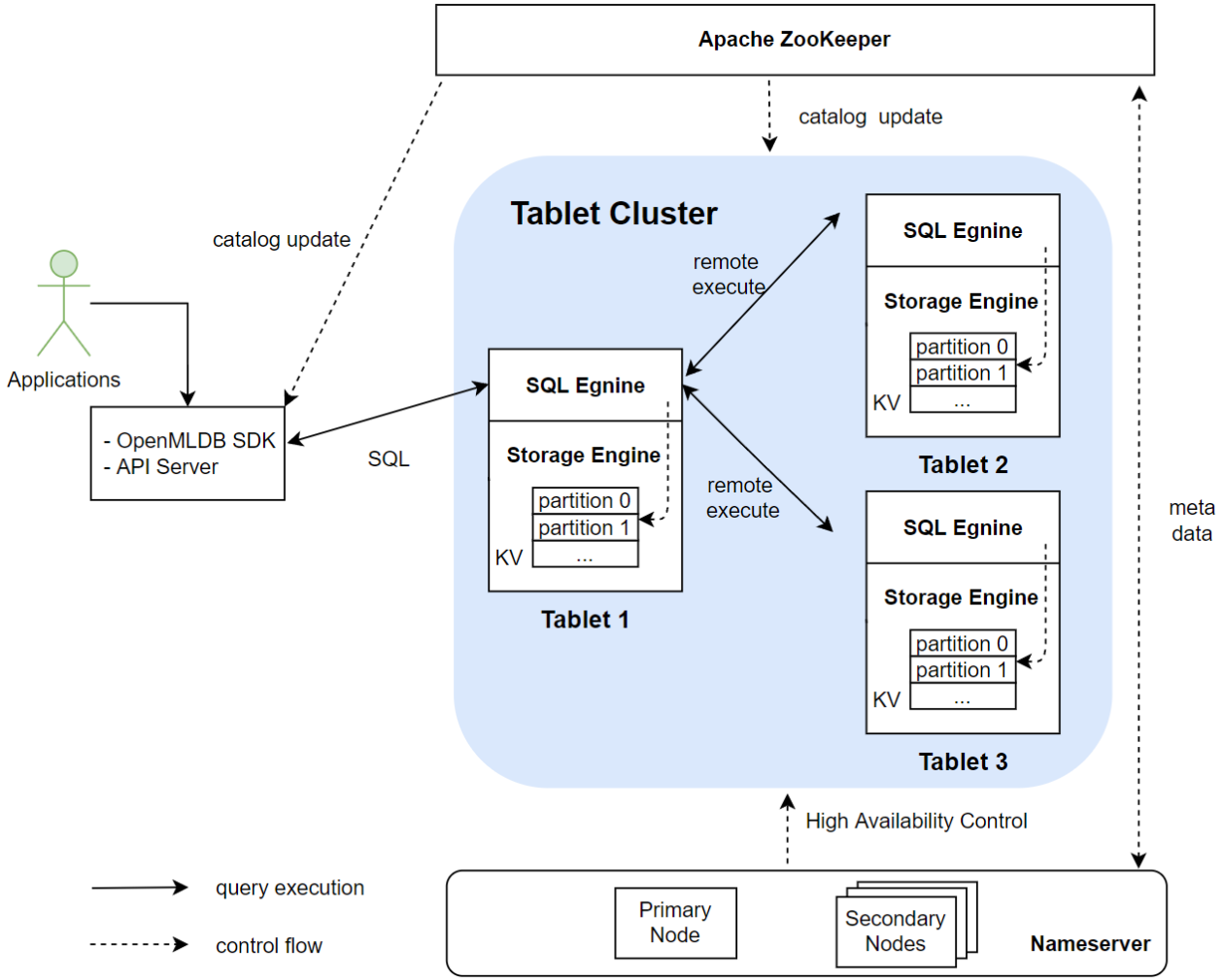
37手游

3. 高可用在线执行和存储引擎

OpenMLDB 线上引擎整体架构

线上引擎包含主要模块

- ZooKeeper – 元数据存储和管理
- Nameserver – Tablet 管理和故障转移
- Tablets
  - SQL 执行引擎
  - 存储引擎
  - 分布式部署



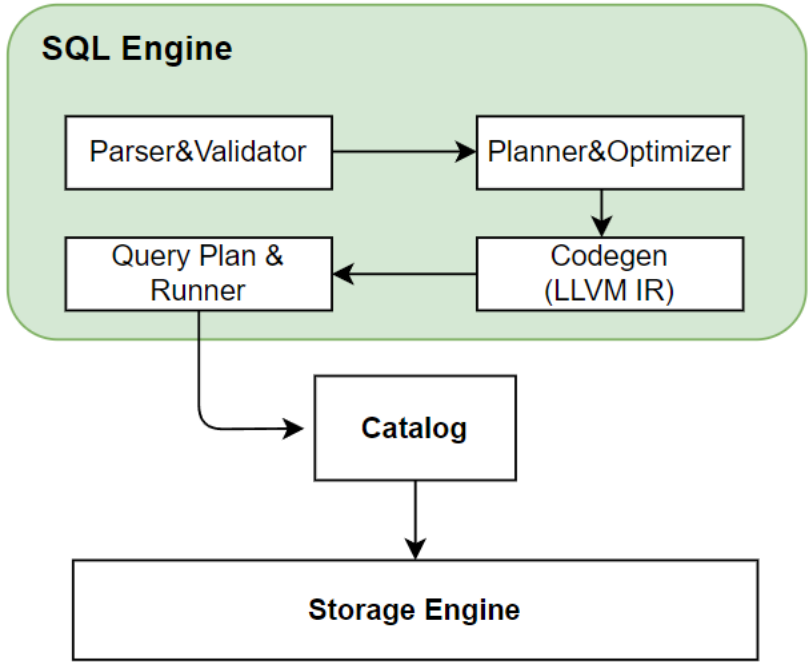
在线 SQL 执行引擎

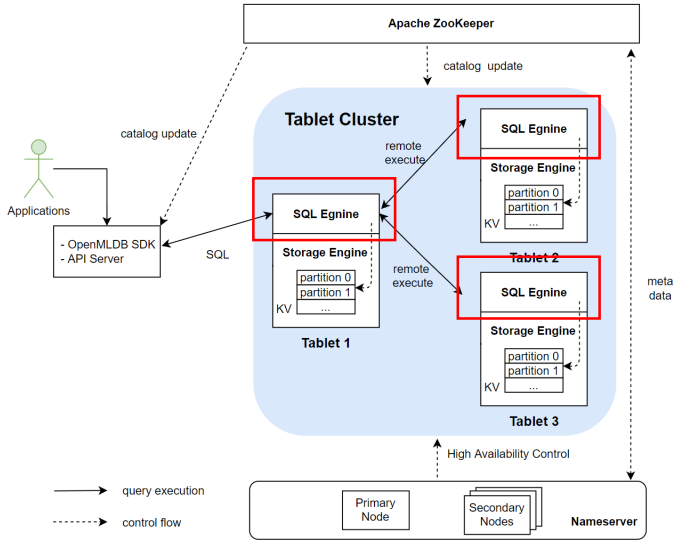
功能强大

- 离线在线统一的 SQL 引擎
- 基于 ZetaSQL 语法扩展
- 支持 UDF 动态扩展

性能优化

- LLVM codegen
- 循环绑定
- 窗口合并

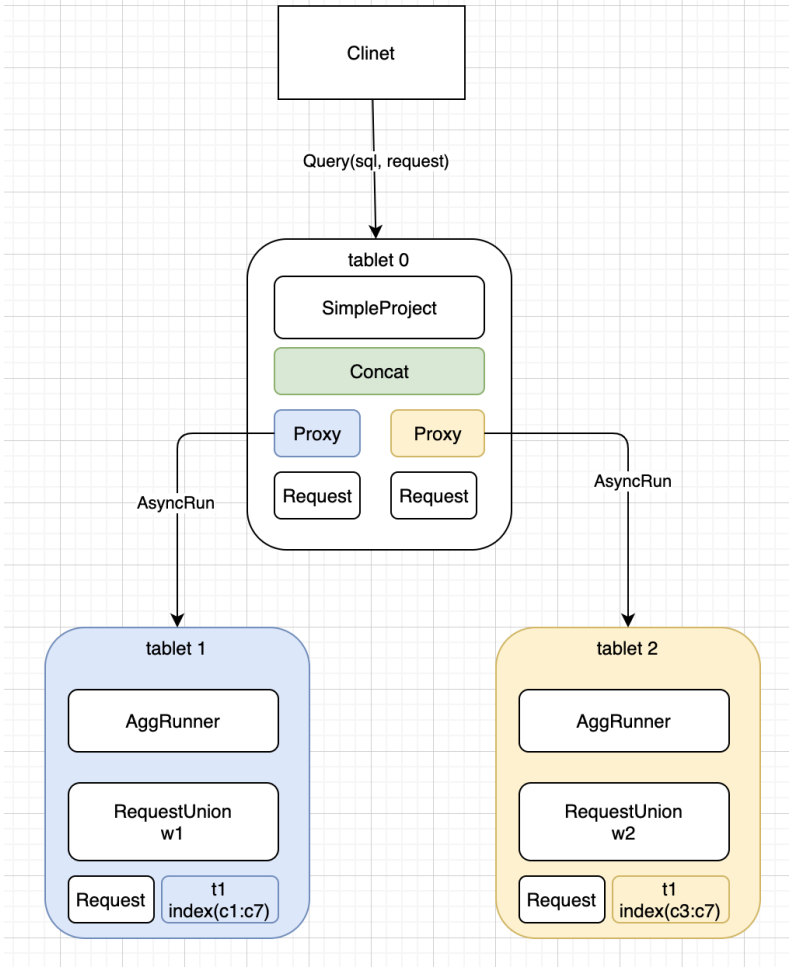


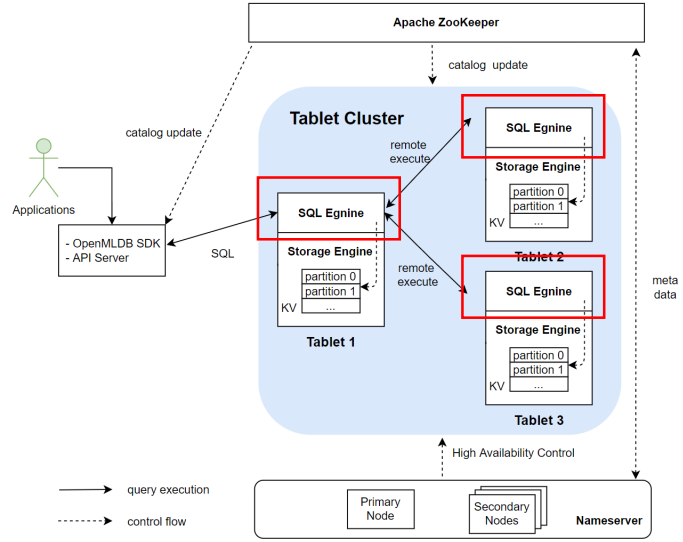


在线分布式 SQL 执行引擎

主tablet作为协调者

- 发送子查询到其他tablets（异步）
- 合并子查询结果
- 子查询并行执行

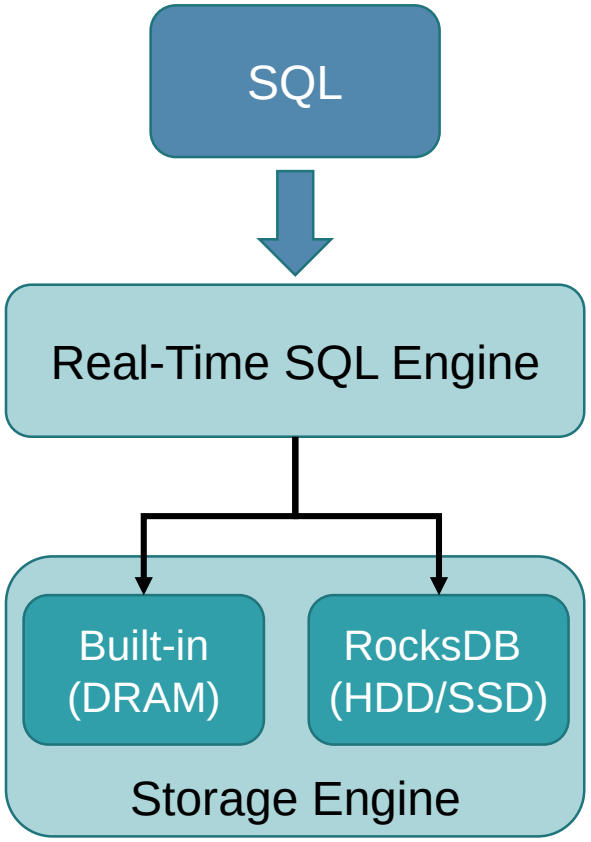


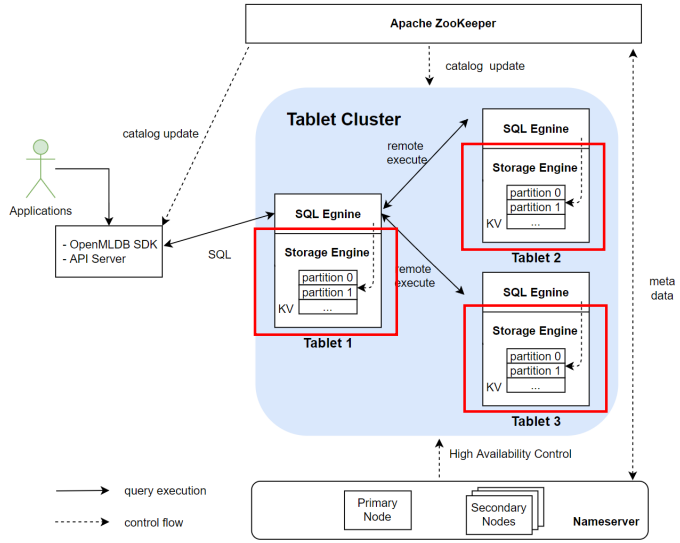


在线存储引擎

在线引擎提供基于内存和外存的两种存储引擎选择

- 基于内存：低延迟、高并发；较高使用成本
  提供毫秒级延迟响应
- 基于外存：对性能较不敏感；低成本使用落地
  基于 SSD 的典型配置下成本可下降 75%
  基于RocksDB开发
- 两种引擎上层业务代码无感知，零成本切换



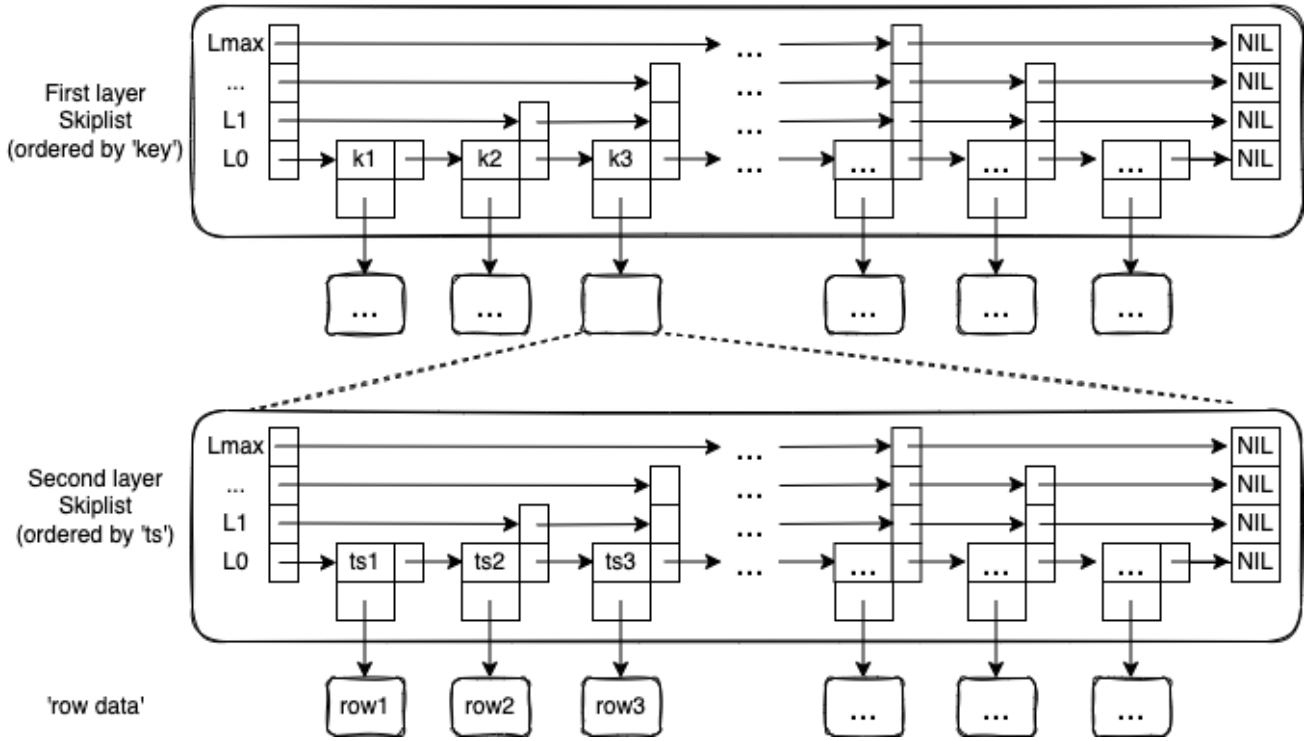



```
CREATE TABLE demo_table1 (col0 STRING, col1 int, std_time TIMESTAMP)
OPTIONS(storage_mode='ssd', replicanum=2, partitionnum=4);
```

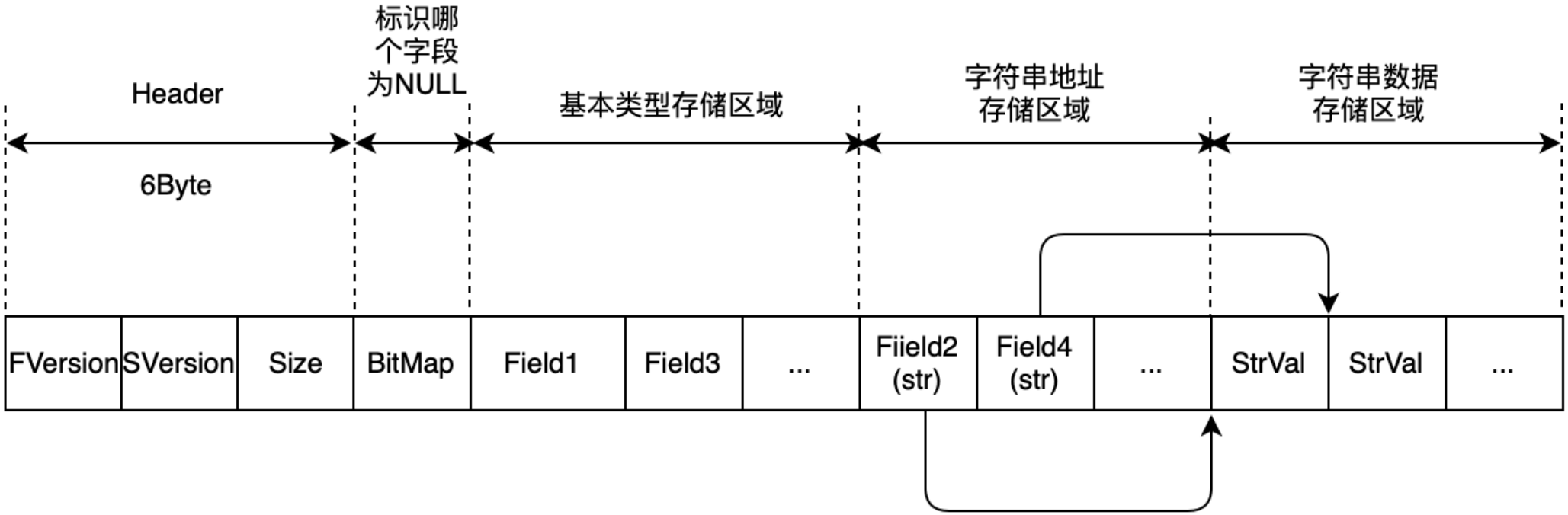
内存存储引擎 – 核心双层跳表数据结构

双层跳表内存索引结构

- 第一层索引对应具体的键值，优化分组操作（如 GROUP BY）
- 第二层索引对应时间戳，高效找到时间窗口
- 高效插入和查询，典型场景时间复杂度 $O(logn)$
- 高效支持数据过期（TTL）相关操作



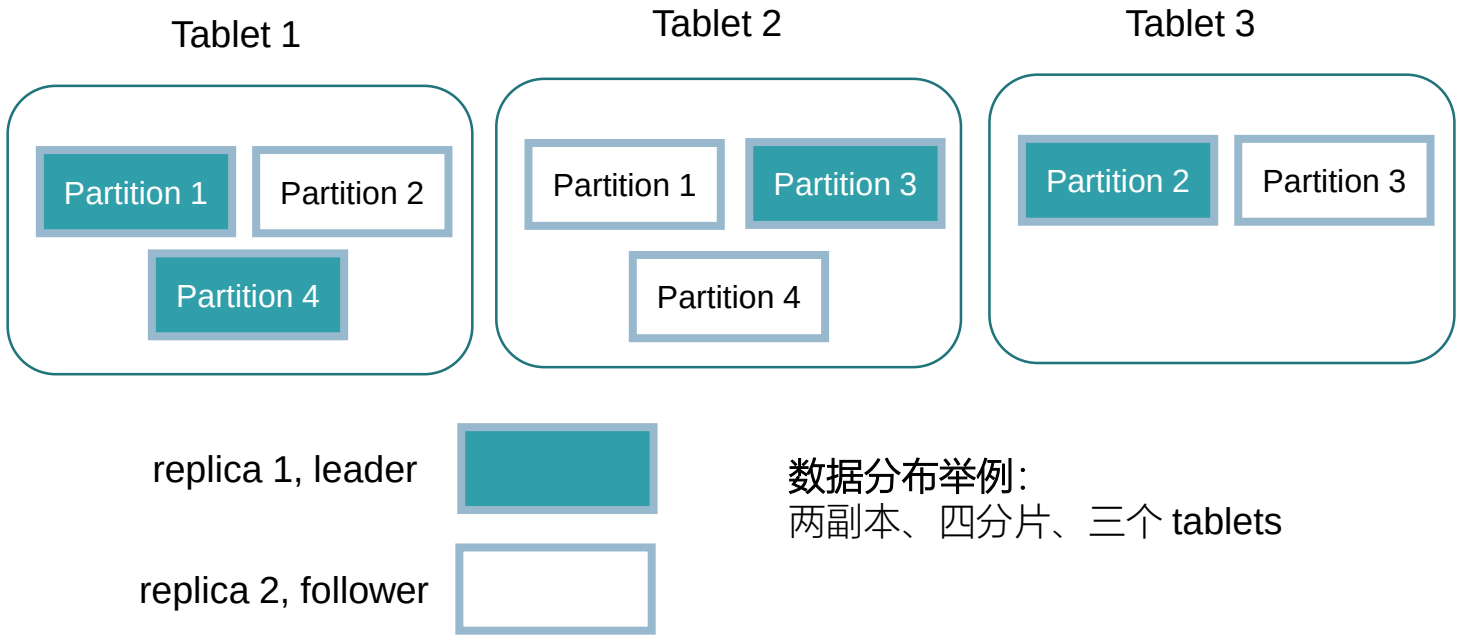
内存存储引擎 – 高效的编解码格式



- 节省内存
- 高效做 projection

内存存储引擎 – 数据分片

- 分片 (partition)
  - 表被切分，提升分布式性能
  - 最小存储单位，分片可以灵活在不同 tablet 之间实现迁移
- 副本 (replica)
  - 表存放多个拷贝，高可用
  - 多个副本的分片会有一个主分片，不同分片的 leader 会分布在不同的节点上，计算请求被发送到 leader
- 系统尽量保证主分片均匀分布，且每个 tablet 保持数量平衡

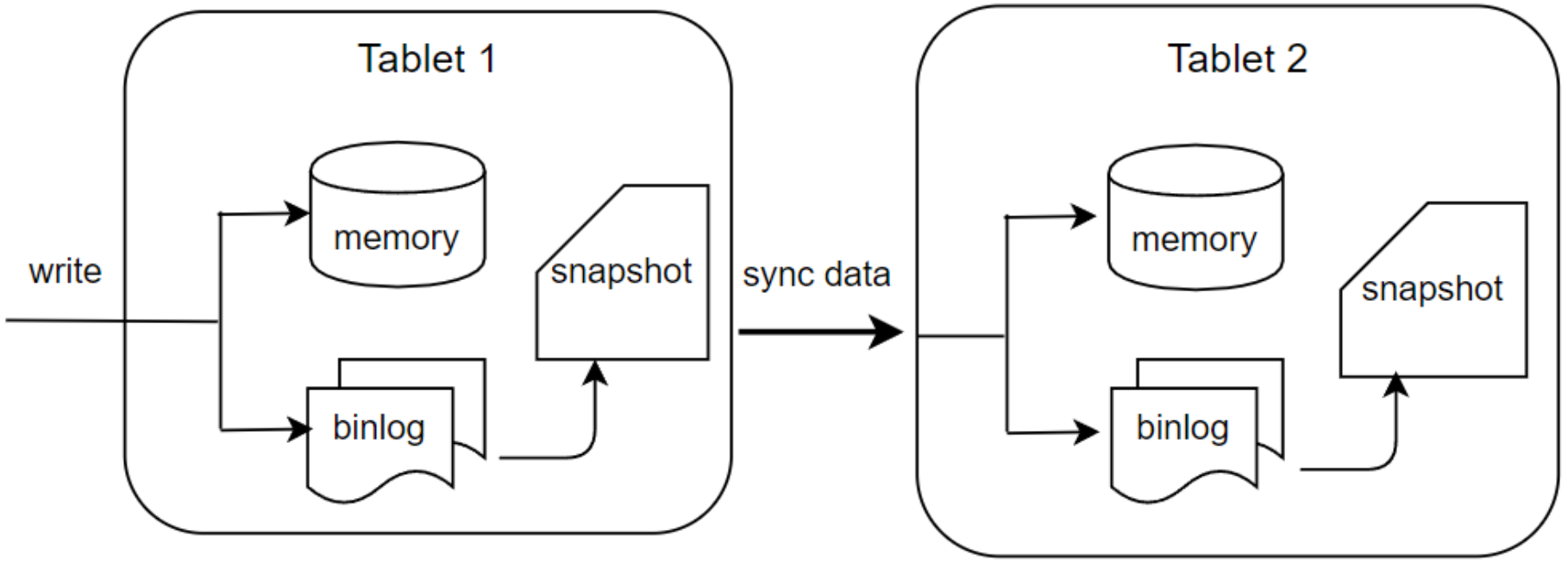


数据分布举例:
两副本、四分片、三个 tablets

```
CREATE TABLE demo_table1 (col0 STRING, col1 int, std_time TIMESTAMP)
OPTIONS(storage_mode='ssd', replicanum=2, partitionnum=4);
```

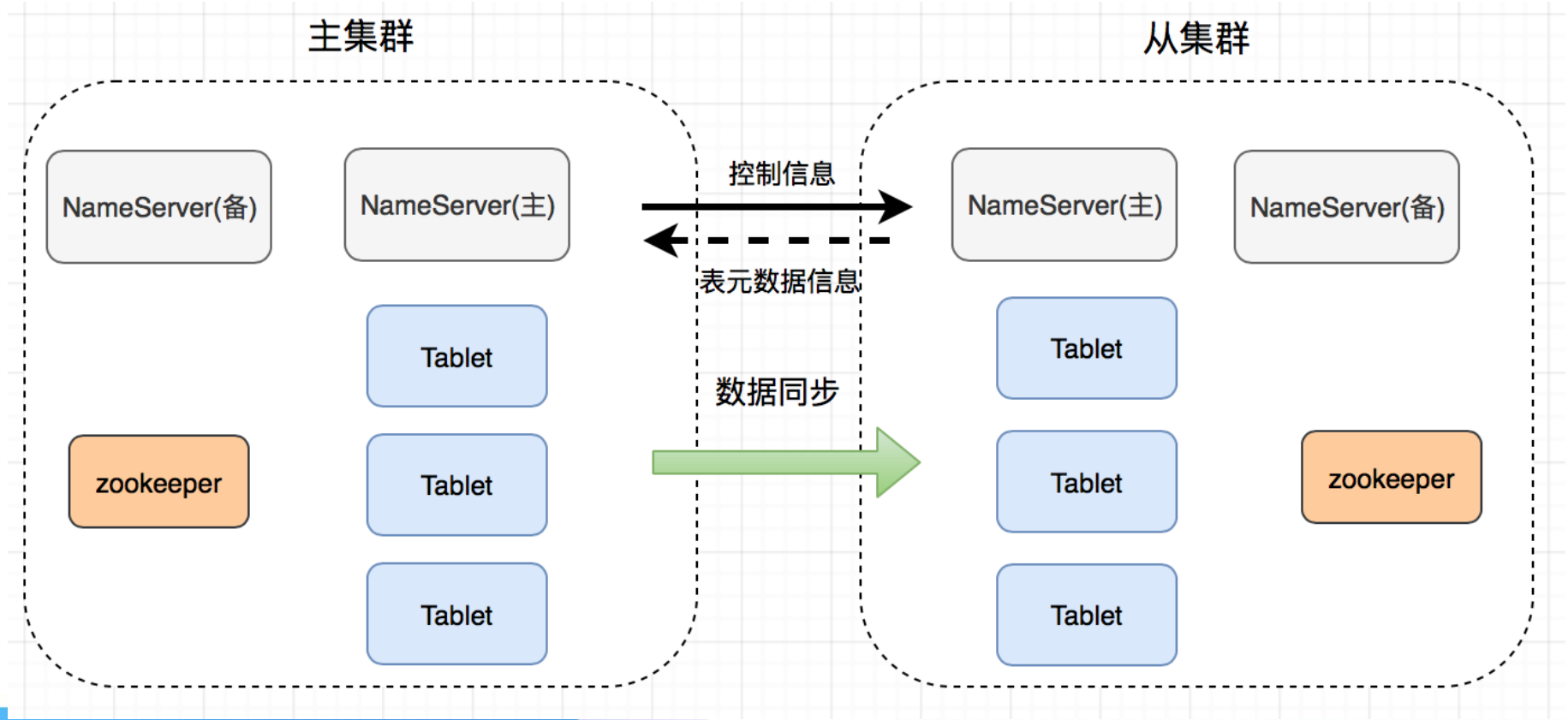
内存存储引擎 – 主从同步

- 定期会生成分片 snapshot
- 数据恢复通过 snapshot + binlog 实现
- 主从之间通过 binlog 实现数据同步



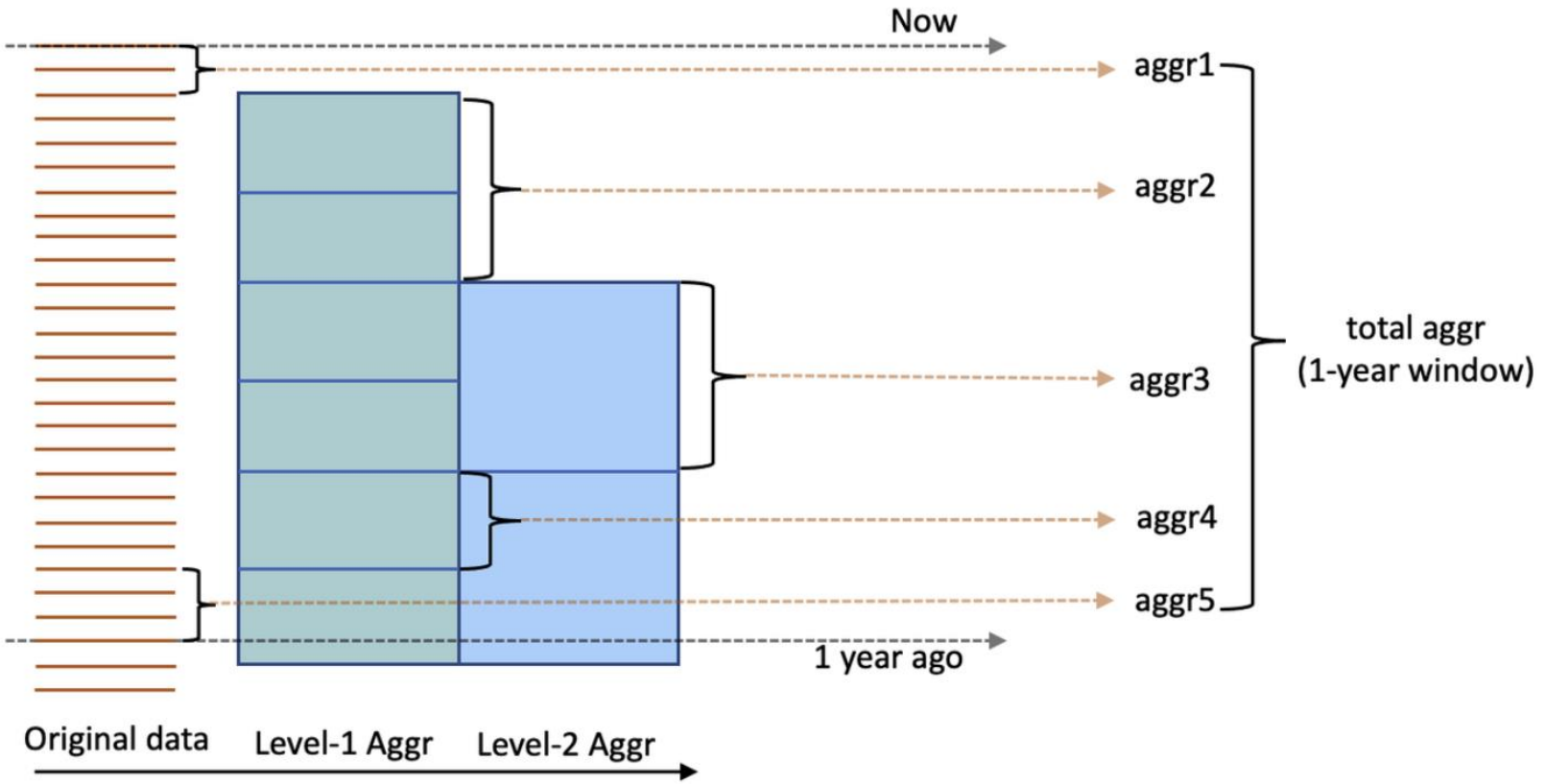
双机房容灾技术架构

主集群：能支持**读写**的集群，并且可以给从集群同步数据
从集群（一个或多个）：只提供读请求的集群，数据和主集群保持一致



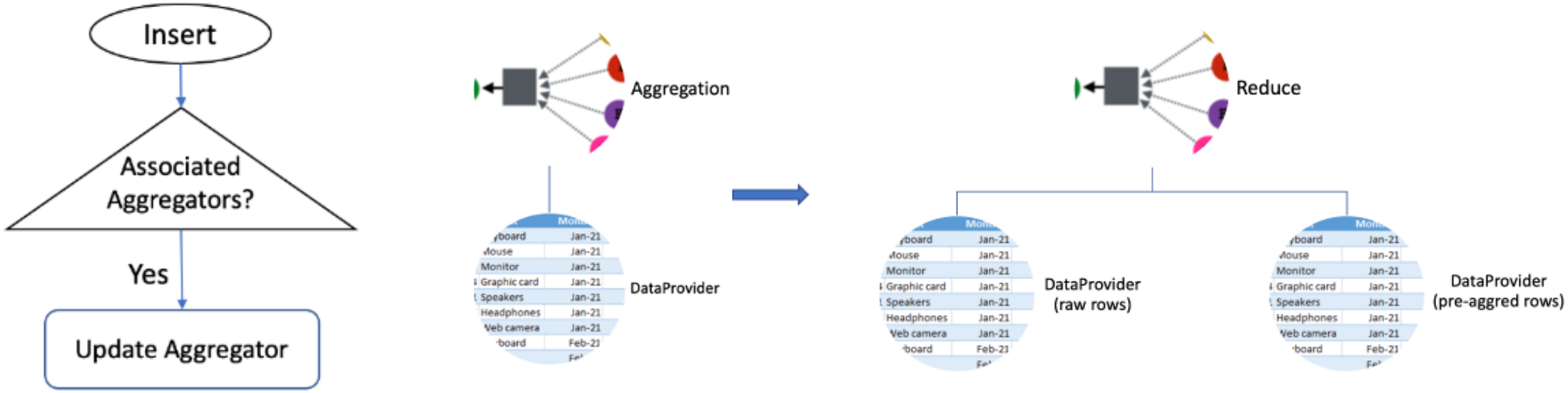
预聚合技术

- 目的：优化长窗口计算效率
- 技术：预聚合部分结果用于实时计算
- 百万数据下提升性能两个数量级
- 同时适用于内存和磁盘存储引擎



预聚合技术

- 预聚合发生在数据插入
- 特征提取：预聚合表 + 原始表 进行聚合



在线引擎性能测试

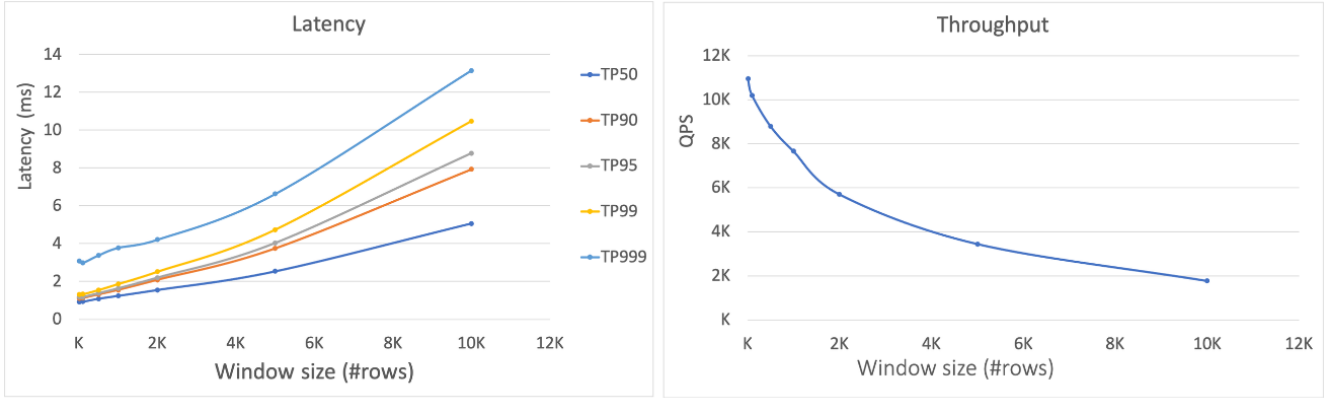


Figure-6: 在窗口内数据条数变化的情况下，延迟（左图）和吞吐（右图）的性能表现

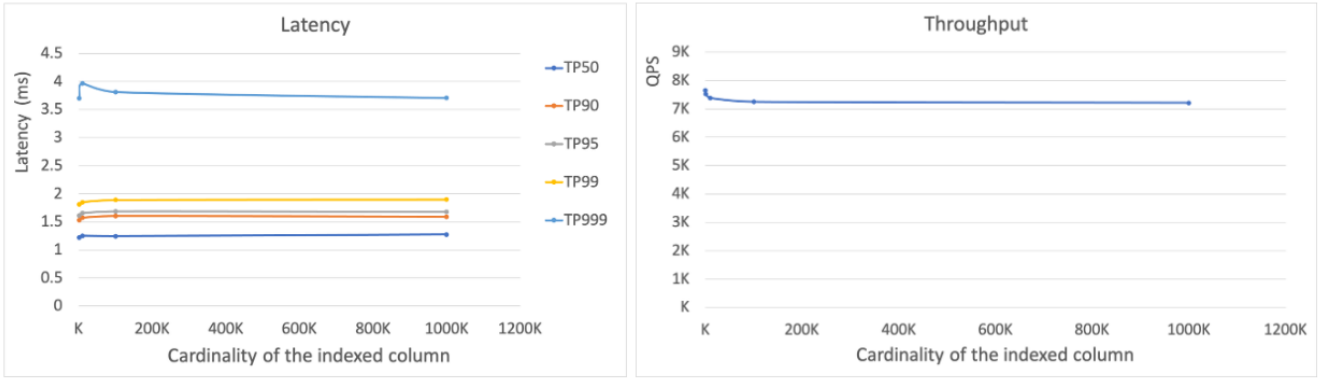
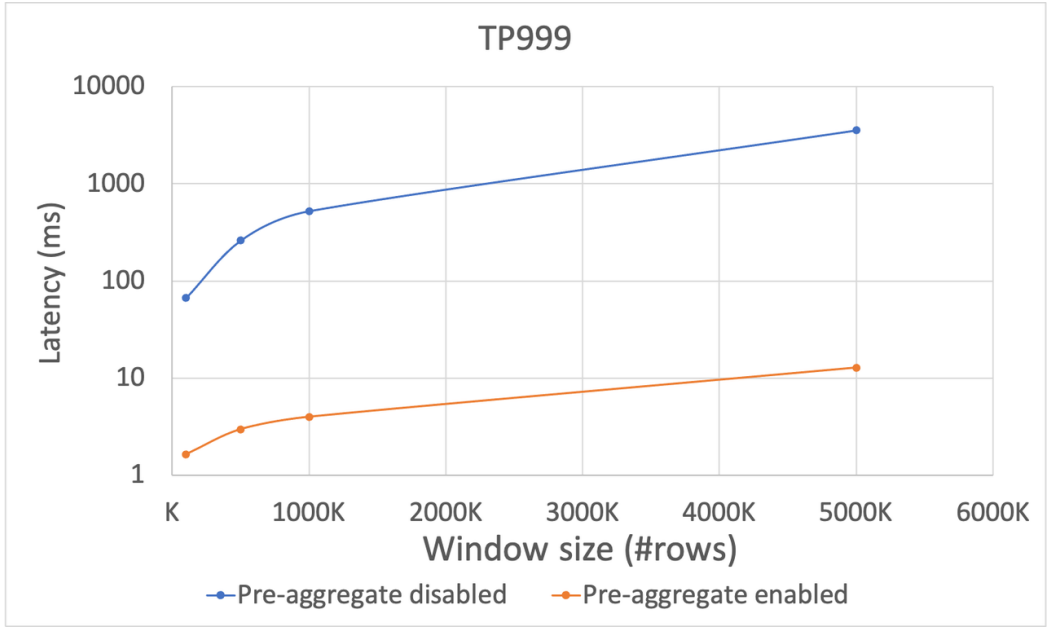
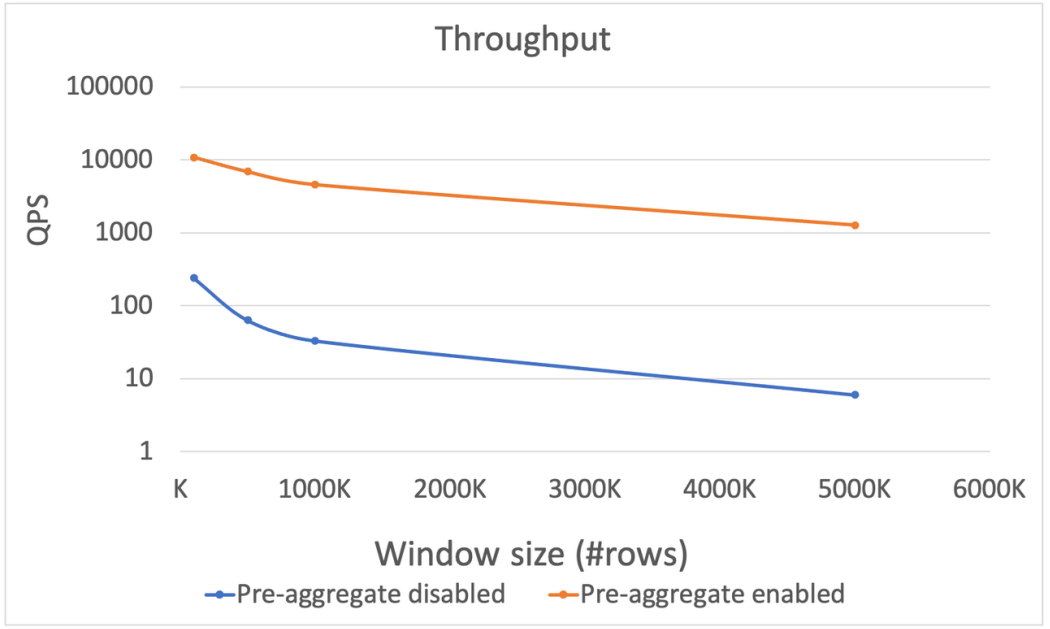


Figure-7: 在索引列基数变化的情况下，延迟（左图）和吞吐（右图）的性能表现

在线引擎性能测试





长窗口预聚合优化使得在长窗口查询下，无论延迟还是吞吐，性能都提升了两个数量级

4. History & Roadmap

OpenMLDB 发展历程

开源前 (RTIDB/FEDB)

- 2017.02 commit第一行代码
- 2017.08 第一个金融客户 -某全国性股份制商业银行
- 2019.09 第一个零售客户 - 某头部餐饮企业
- 2020.12 落地超过100个场景

2021.6

OpenMLDB 正式宣布开源

2021

- 7月 0.2.0 版本 发布，支持RestAPI访问
- 8月 基于 OpenMLDB 的优化创新论文在国际顶级数据库学术会议 VLDB 2021 发表
- 获得Gitee认可，荣获GVP称号
- 9月 第一个开源企业用户 Akulaku
- 荣获信通院"开源社区及开源项目"奖项
- 11月 0.3.0版本 发布，支持单机模式
- 12月 OpenMLDB贡献者计划启动
- 举办国内首次Feature Store Meetup

2022

- 1月 0.4.0 版本 发布，优化 SQL 体验
- OpenMLDB 第一次Meetup
- 5月 0.5.0 版本 发布，支持预聚合，UDF
- 8月 0.6.0 版本 发布，引入智能诊断工具

OpenMLDB v0.7 Roadmap

SQL 能力

- List, map
- Lambda functions
- Built-in functions

稳定性

- 内存资源隔离
- 报警系统完善

易用性

- 运维命令优化
- 日志优化
- 阿里云上线

欢迎加入 OpenMLDB 开源社区

OpenMLDB 中文官网：https://openmldb.ai/
GitHub: https://github.com/4paradigm/OpenMLDB

OpenMLDB 微信交流群

WE ARE HIRING
(Fulltime & Intern)

个人微信（张浩）

THANKS