快手数据平台 | 快手中学 KUAISHOU SCHOOL | 快手项目管理

如何使用Hudi解决效率问题

靳国卫

快手大数据研发专家

自我介绍

靳国卫　快手大数据研发专家

- 负责用户增长数据团队
- 着重OLAP技术和产品、数据内容领域建设方向探索

01 痛点–业务场景

数据调度

- 业务周期非自然周期**【启调晚】**
- 局部**【更新】**部分数据，需要**【动态回刷】**数据
- **【启调晚】** **【大范围回刷】** **【计算时间长】** ➔ **【就绪晚】**

数据同步

- **【同步】** **【大量】** **【过程明细】**数据到数仓
- 需**【最新状态】**结果，因此**【合并计算】** **【计算时间长】**
- **【合并计算】** **【计算时间长】** **【根节点】** ➔ **【3点】**就绪，SLA保障压力大

修复回刷

- **【回刷】**业务调整的历史数据
- 仅**【更新】** **【局部小部分】**数据即可
- 当前技术方案**【回刷】** ➔ **【修复慢】** **【周期长】**

01 痛点场景–思考

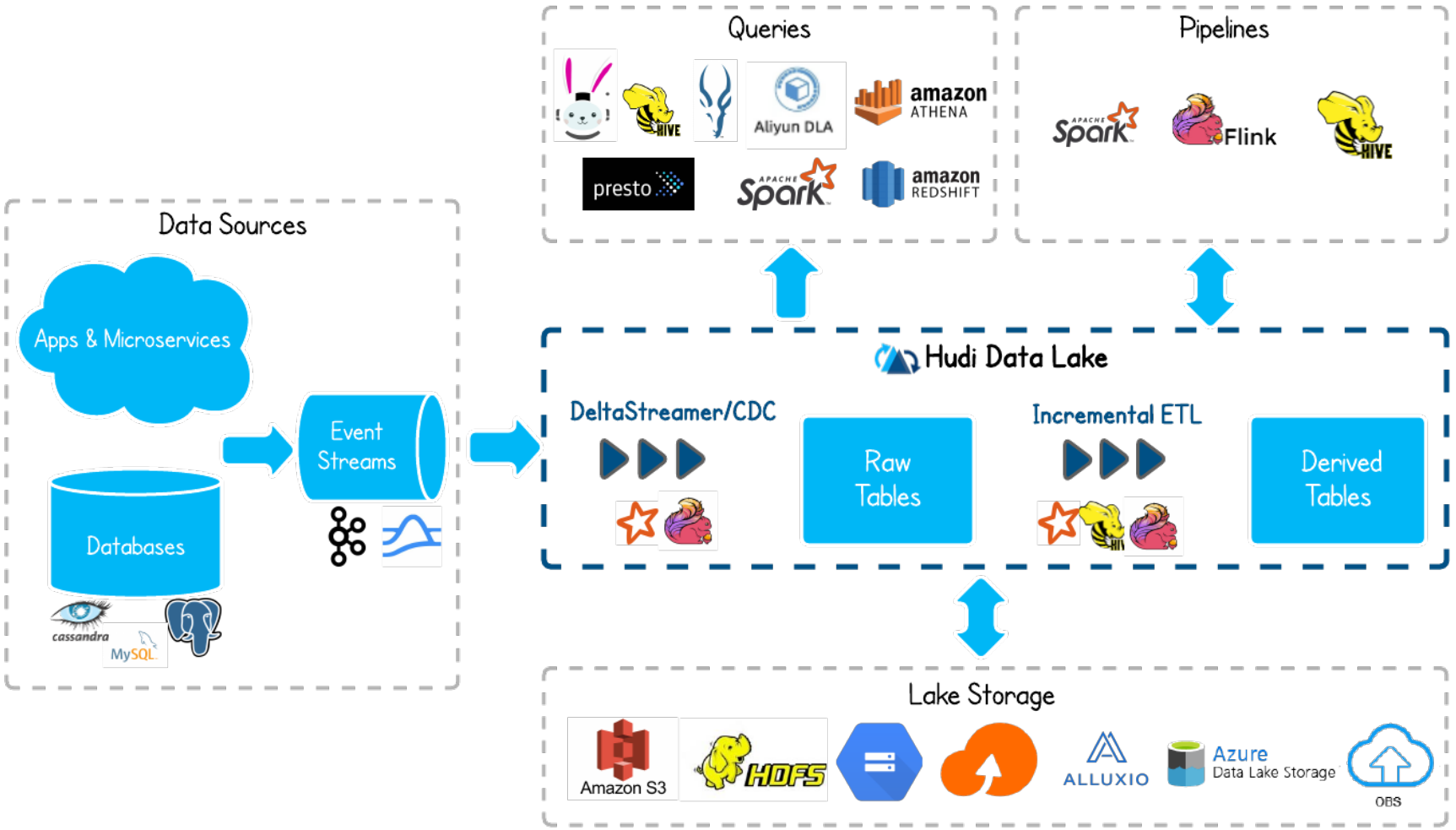
场景	痛点诉求	当前方案	更新特点 (总体占比小)	单方向思考	单方向解决思路
数据调度	产出快 状态新	小时调度 动态回刷	总体量中 更新量小	实时化	Flink、天窗口sink
数据同步	产出快 状态新	小时同步 天Merge	总体量大 更新量大	优化 实时化	小时同步 小时Merge
修复回刷	产出快 状态新	数据回刷	总体量大 更新量小	优化 执行快	

OLTP(CRUD)
实时化 + 大数据的CRUD

02 方案：技术选型（实时化、离线CRUD）

对比项	Hudi	Delta Lake	Iceberg
功能丰富度	丰富	丰富	待提升
公司融合度	高		
自动化程度	自动	部分开源	手动
Flink集成	是	否	是
社区活跃度	活跃	活跃	相对活跃

02 方案：Hudi为何能解决痛点



EventStreams ->

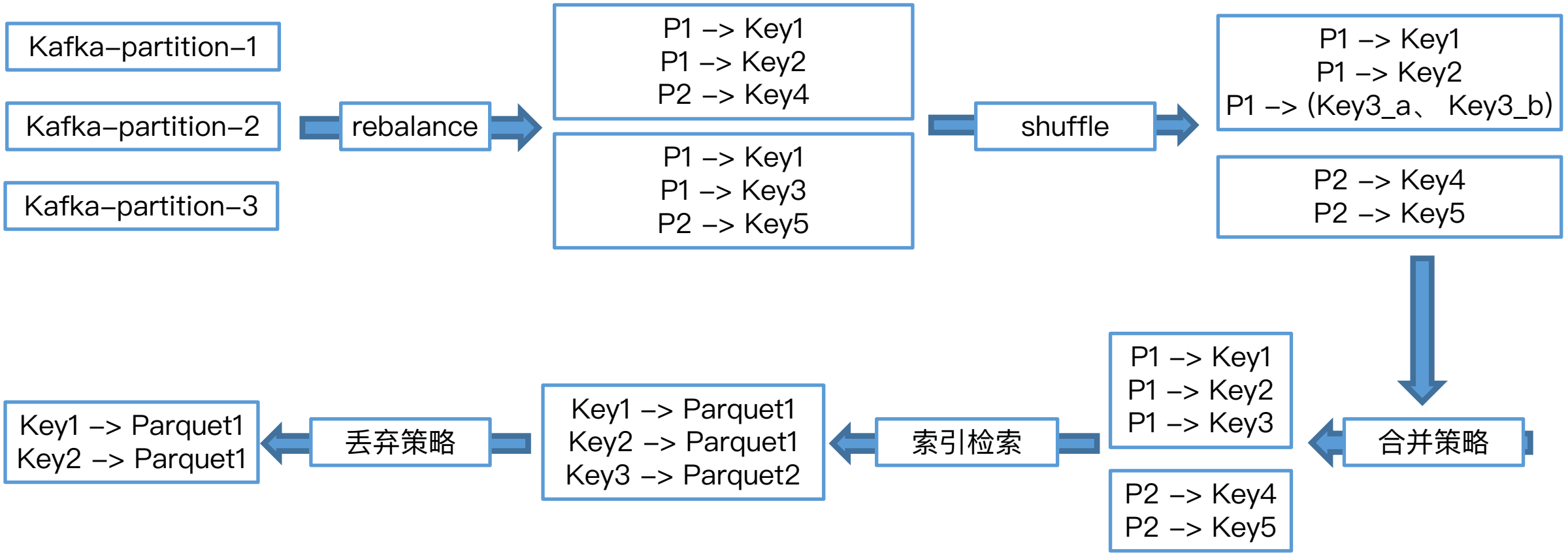
DetlaStreamer

就绪快、实时化

Row Tables

状态新、离线(CRUD)

02 方案：一条数据的Hudi之旅(计算存储过程)



02 方案：一条数据的Hudi之旅(计算存储结果)

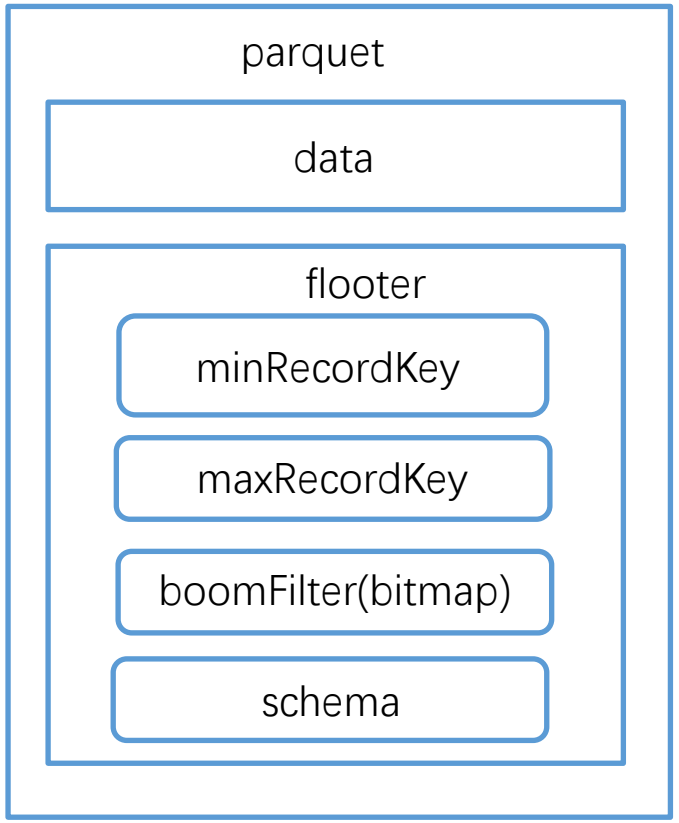
➢ **存储目录结构**

- viewfs://Hadoop/../base_active_device_di/SEM/20210501/55/..8aa6-df0d9cf3c0d6-2_20201026210468.parquet
- viewfs://Hadoop/../base_active_device_di/SEM/20210501/55/..8aa6-df0d9cf3c0d6-2_20201026210475.parquet

➢ **存储目录拆解**

- BasePath： viewfs://Hadoop/../base_active_device_di
- PartitionPath：SEM/20210501/55
- FileID： ..8aa6-df0d9cf3c0d6-2
- CommitID : 20201026210468

数据存储方案
整合计算过程的引擎



03 痛点–业务场景回顾

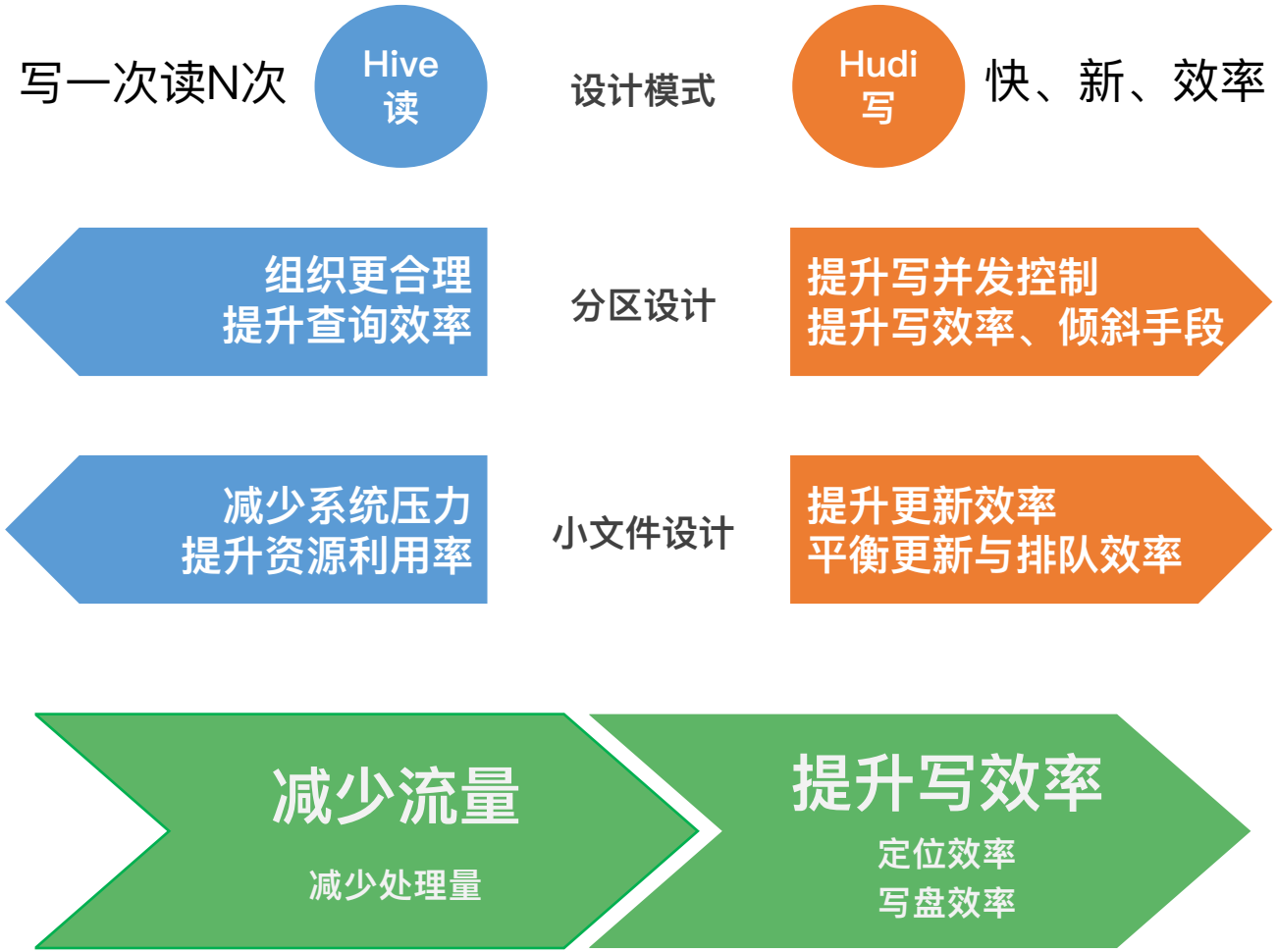
数据同步

- **特点：** 数据量大、计算时间长、大量数据大更新
- **痛点：** 时效性差、SLA压力大

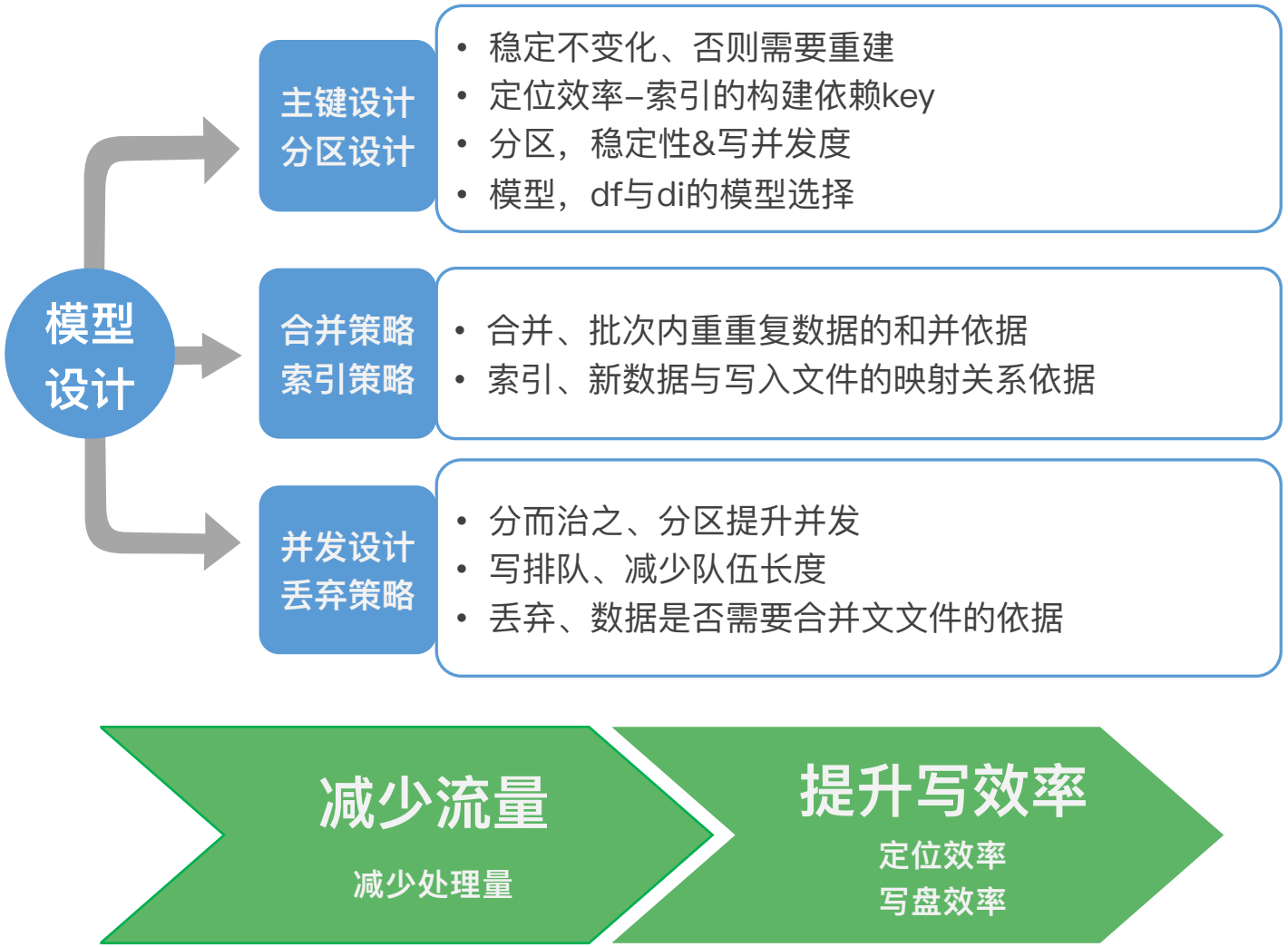
修复回刷

- **特点：** 数据跨周期长、大量数据局部新
- **痛点：** 修复慢、大量资源浪费

03 挑战1：基于Hudi的模型设计（认知偏差）



03 挑战1：基于Hudi的模型设计（写模型设计）



03 挑战1–基于Hudi的模型设计（实战）

数据同步

- **【数据同步】**遇到**【海量】** **【过程】**明细数据
- 诉求**【最新状态】**关系，大量**【合并计算】**
- **【合并计算】** **【计算时间长】** **【根】**同步**【3点】**左右就绪，SLA保障压力大

模型设计：

模型：最新关系数据，DF模型
主键：user_id、follow_id
分区：product/mod(关系)
文件：256m
索引：布隆索引
合并策略：保留最新
丢弃策略：丢弃历史

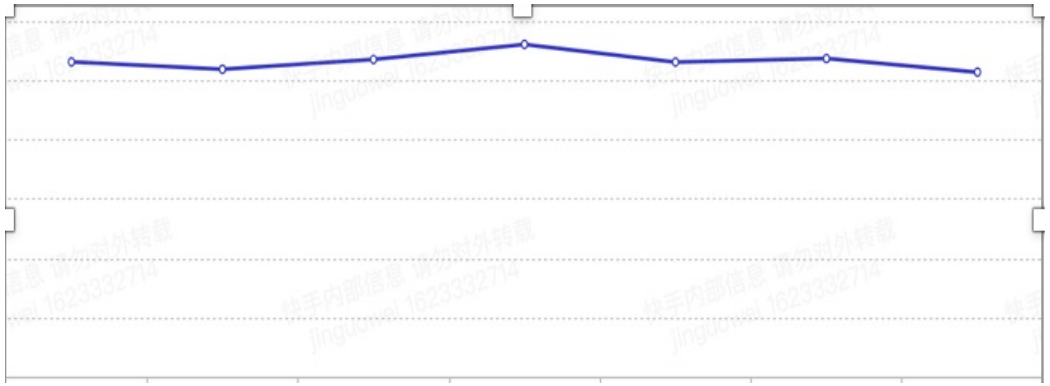
数据现状：

总量：千亿级
存储：10T+
每天更新：亿级

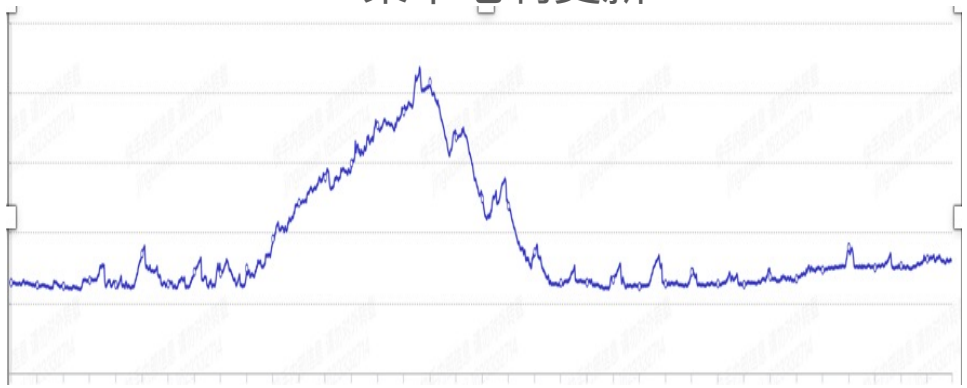
03 挑战2–集中毛刺更新（写失败）

场景：快手经常在节假日、大V活动时会出现大量的关系数据；这时候就会出现一段时间业务突增

平滑更新



集中毛刺更新



问题：

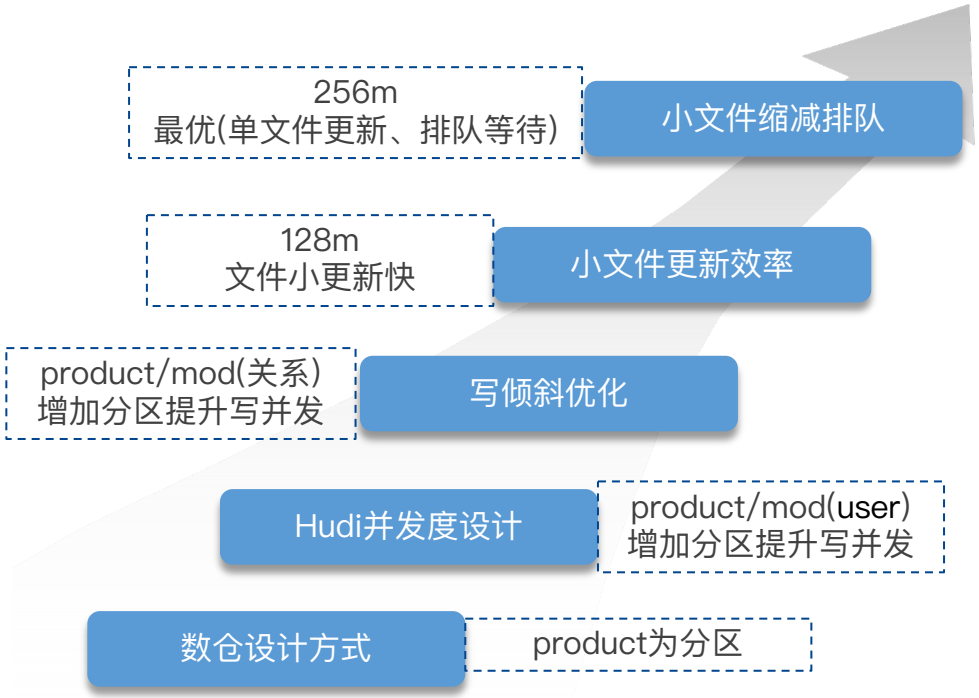
1、数据峰值任务处理周期变长、引起时效性**报警**

2、数据量持续变大引起程序GC严重、任务OOM等问题造成**任务失败**

3、峰值持续时间长、造成数据积压，失败的任务**重启失败**

03 挑战2：集中毛刺更新（解决）

写慢！提升并发、Sink效率



定位慢！合并、索引、丢弃策略优化

合并&丢弃策略

1、关注后立即取消，10%数据减少

2、丢弃部分数据，减少重复、乱序的写盘

布隆索引

1、数据与文件映射关系存储在文件

2、任务失败时重新读文件即可，效率也不错

3、单节点仅加载单分区的索引资源可控

03 挑战3：数据回刷局部更新（研发效率）

场景诉求：数据修复回刷

1、按天进行业务分析

2、能够实现【局部历史更新】效果

3、沉淀一套通用解决方案

日期	设备	产品	首渠道	金额	修正成
20210503	d1	K	SEM	200	PinPai
20210503	d2	K	PinPai	50	----
20210502	d1	K	SEM	100	PinPai
20210501	d1	K	SEM	300	PinPai

模型设计：

模型、按天分析的场景、di天增量设计

分区：p_date、product

主键：设备

索引：按分区更新使用【布隆索引】

合并策略：最新版本

丢弃策略：历史版本

小文件：128m

03 挑战3：数据回刷局部更新（研发效率）

解决方式：

1、离线数据直接生成Hudi文件Load到表

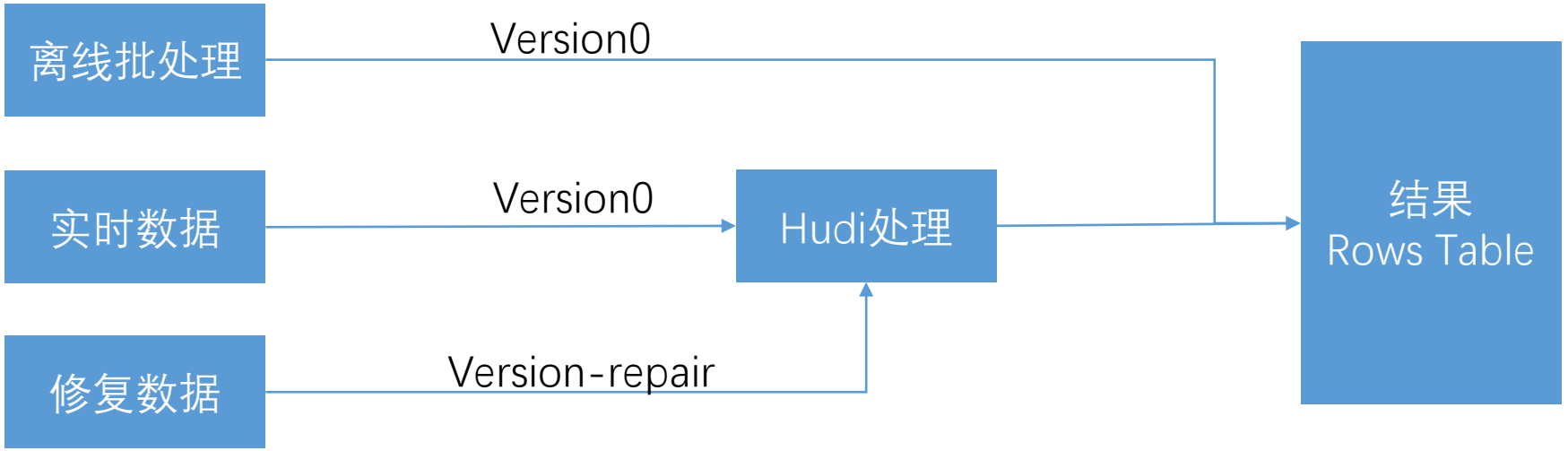
2、实时接收数据，通过Hudi的Real功能生成表

3、日常数据使用模型合并&丢弃版本

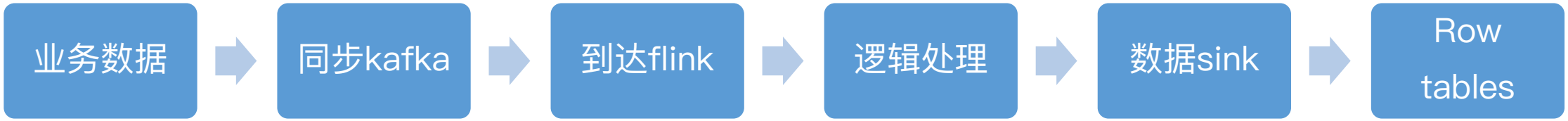
4、修复数据使用新版本，比如修复日期

5、通过版本对比更新达到数据修复的结果

日期	设备	产品	首渠道	金额	版本
20210503	d1	K	PinPai	200	20210503
20210502	d1	K	PinPai	100	20210503
20210501	d1	K	PinPai	300	20210503



03 挑战4：Hudi保障



设计流程

- 模型设计
- 流程审查

时效性

- 关键识别
- 监控预警

准确性

- 探针采样
- 任务阻断

03 成果—成效效果

提升时效

- 花费数据调度计算从**3小时**缩短到**10分钟**内
- 关系数据同步就绪时间从**03:00**提前到**00:30**分
- 数据修复回刷周期从**一周**缩减到**1天**内

资源节省

- 花费&关系场景只要存储最新全量终态数据，节省**3PB**资源
- 设备首渠道只需要更新数据，不需要回刷节省**50T*7**天资源

沉淀推广

- 形成了一套通用的Hudi使用方案，提升了研发效率
- 总结沉淀应用场景后，推广到增长、活动方向去解决业务痛点