

# Unique Key 写时合并 与多源 Catalog 的探索实践

李昂

高级大数据研发工程师

Apache Doris Contributor

# 目录

1. 数仓探索之路
2. Doris 在 C 端的实践
3. 多源 Catalog 推动流批一体的实践
4. 架构演进的经验与总结
5. 未来发展规划

# 1 数仓探索之路



## 业务背景



### 阶段一: 支持内部数据分析

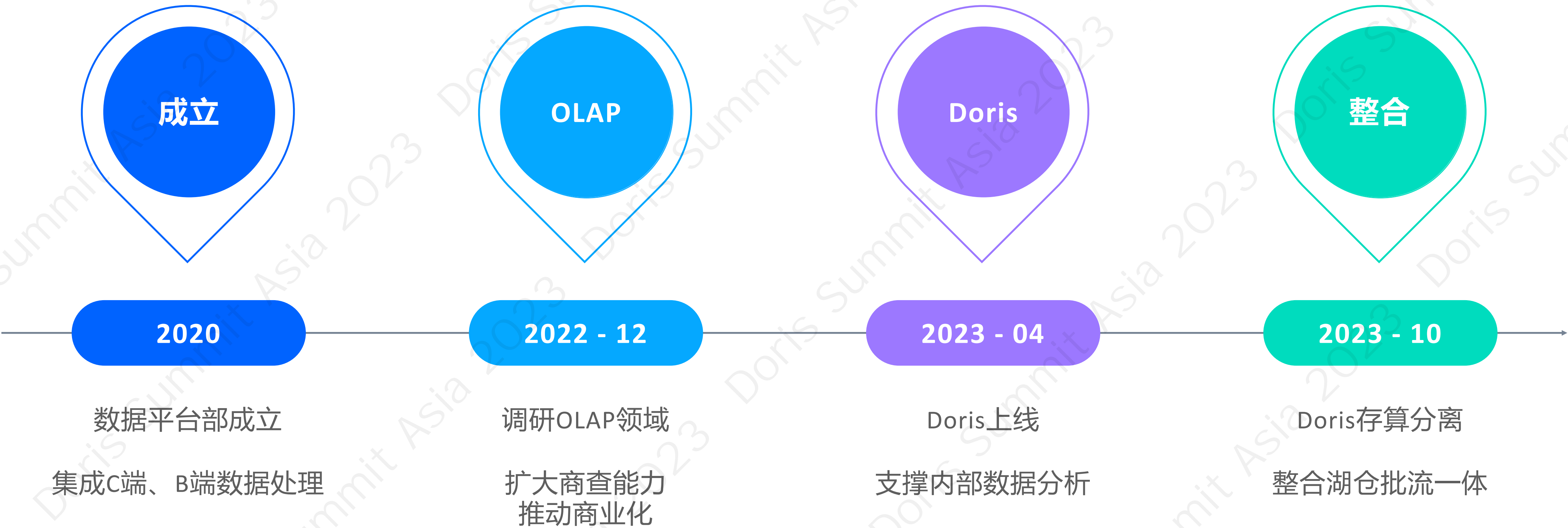
1. 支持人群包圈选与优惠券发放计算
2. 支持订单分析与推送信息分析



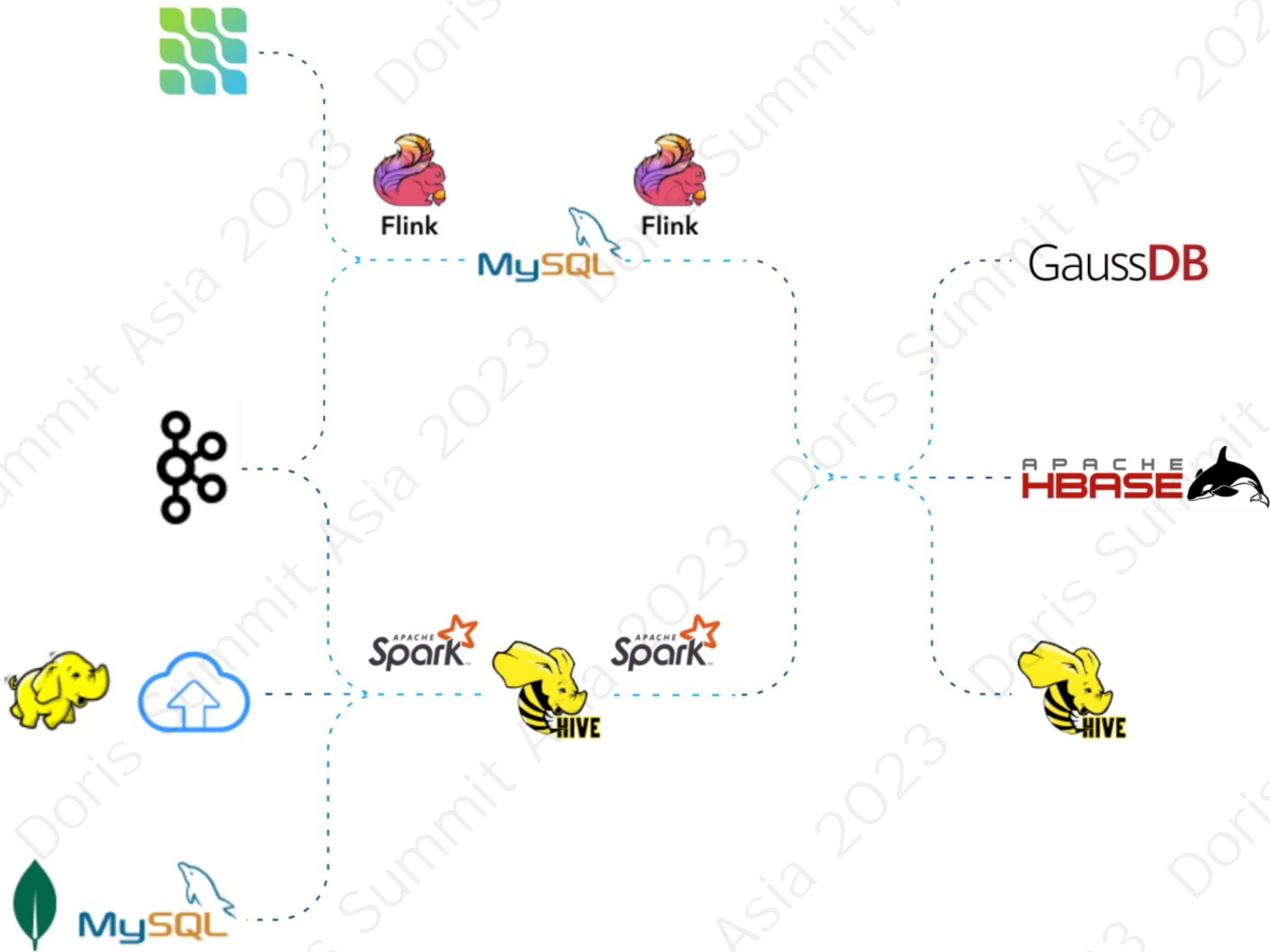
### 阶段二: 提高C端商查能力

1. 提升商查数据内部分析能力
2. 丰富 C 端图形化,提供多维联合展示

数据平台演进



# 架构 1.0



## 实时离线分离

- 1. 实时链路负责增量计算, 保证C端时效性
- 2. 离线链路负责存量计算, 提供内部看板、B端交付能力

## Lambda架构

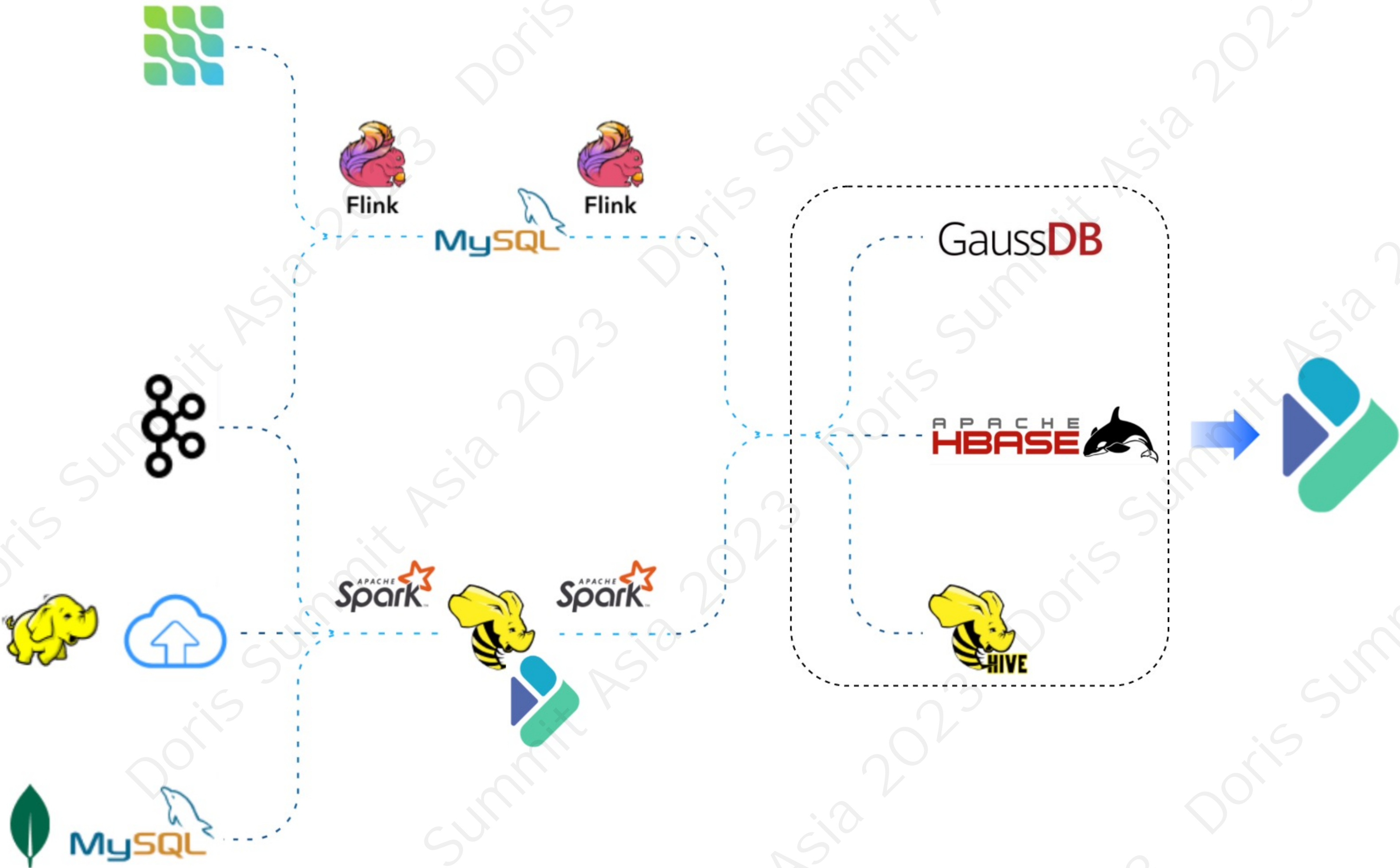
- 1. 链路冗余
- 2. 开发周期长
- 3. 维护成本大
- 4. 需求迭代慢

选型调研

候选引擎 比对项	Doris	Clickhouse	Greenplum
标准SQL支持	支持	支持有限	支持
人力成本投入	轻量级搭建, 周边工具齐全 成本较低	依赖ZK, 扩容复杂 需要额外维护成本	底层是PostgreSQL 需要学习成本
并发查询性能	较高	有限	较高
数据一致性	Exactly-once	At-least-once	Exactly-once
实时upsert、delete性能	高	一般	高
支持场景	丰富	较为单一	丰富
社区活跃度	高	较高	较高



# 架构 2.0

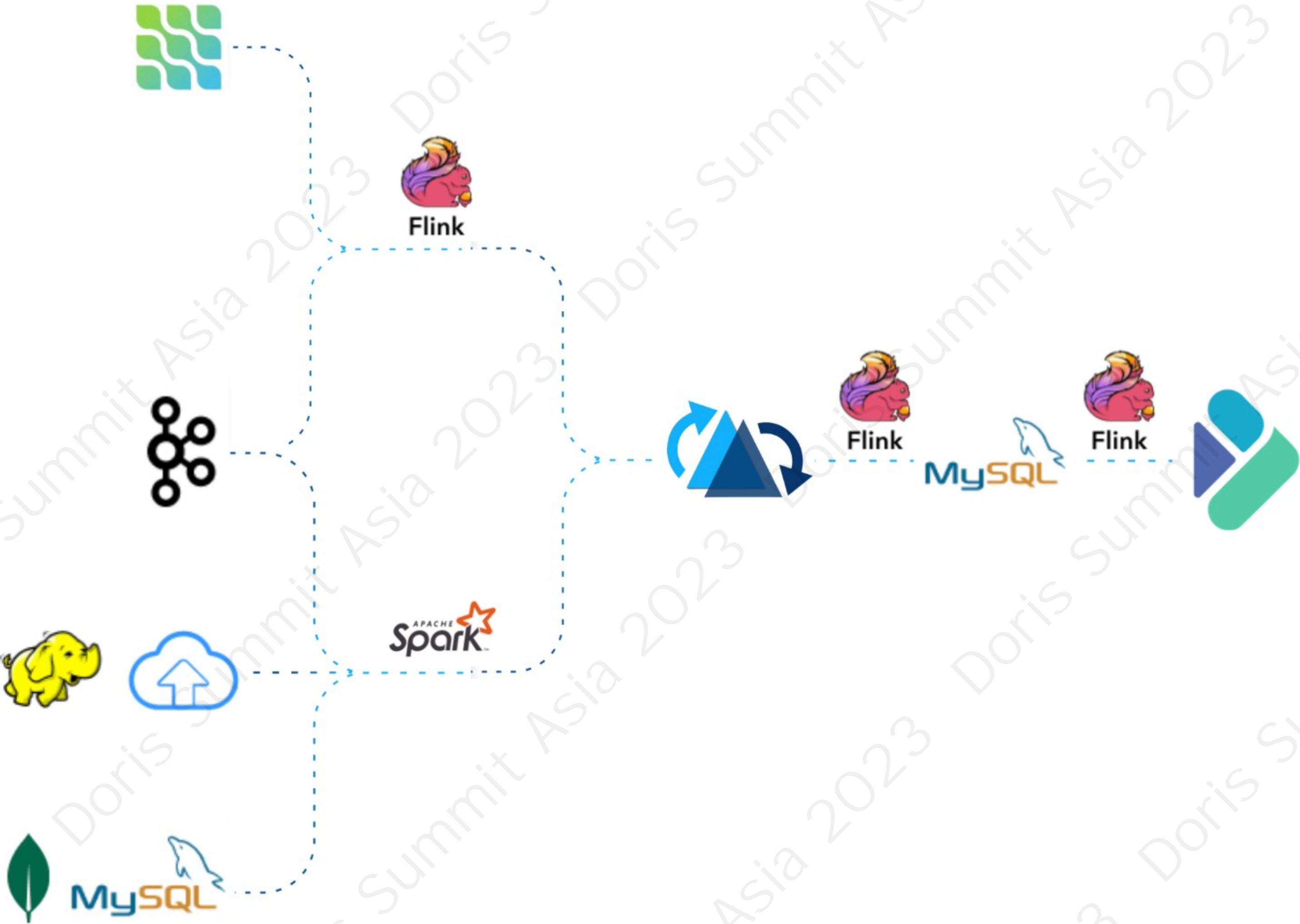


## All In One

- 1. MPP优秀的即时查询与即时更新能力
- 2. Agg模型replace\_if\_not\_null提供partial-update能力, 提升宽表开发效率
- 3. 丰富join计划加速联邦查询
- 4. MOW&多副本, 支持C端高并发
- 5. Light Schema Change提升结构扩展能力
- 6. 低替换成本



# 架构 3.0



## 多源Catalog促进链路流批一体化

### 一. 存算一体

- 1. 商查数据表落地Doris, 最大限度降低查询时传输开销, 响应C端线上高并发

### 二. 存算分离

- 1. 日志数据、冷数据存储在Hudi外表, 降低磁盘冗余, 提升Doris节点IO计算利用率
- 2. 减少后台tablet维护开销, 提高集群稳定性
- 3. 节点扩容更加高效, 降低维护成本

## 2 Doris 在 C 端的实践

# 多模型、多功能支持

## Merge-on-Write

简单数据模型, 开启unique表的  
MOW功能, 提升并发查询性能

## Replace-if-not-null

通过agg表的partial-update能力  
简化宽表代码逻辑, 提升开发效率

## Colocate-join & Bucket-join

加速的join执行效率,  
补齐未预聚合短板

## Light Schema Change

秒级结构变更, 数据扩展更灵活





## Merge-on-Write

```
CREATE TABLE `company_base_info` (  
  `company_id` bigint(20) NOT NULL,  
  `company_name` string NOT NULL  
) ENGINE=OLAP  
  
UNIQUE KEY(`company_id`)  
  
DISTRIBUTED BY HASH(`company_id`) BUCKETS 32  
  
PROPERTIES (  
  "replication_allocation" = "tag.location.default: 5",  
  "enable_unique_key_merge_on_write" = "true",  
);
```

单点公司查询速度提升

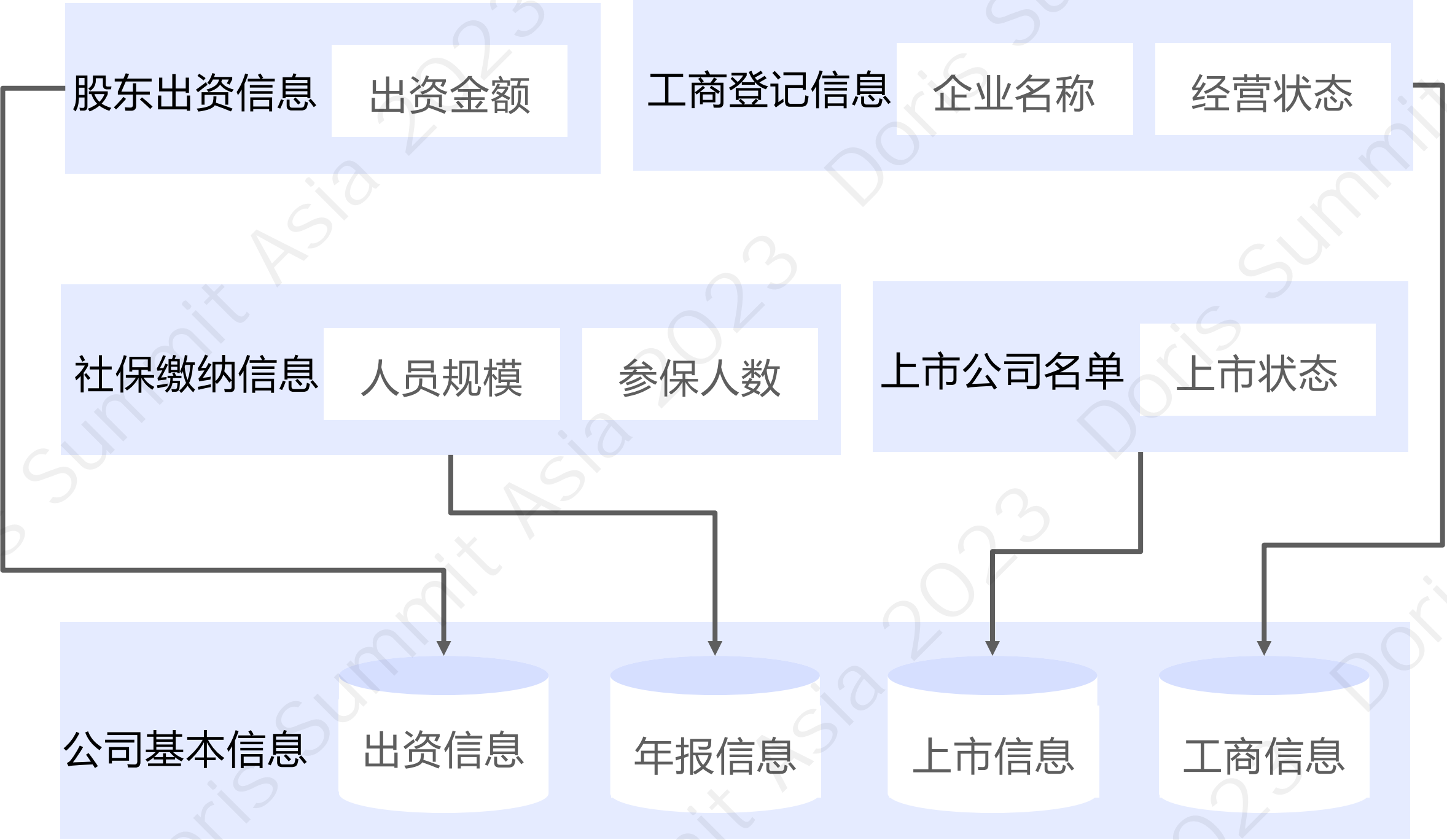
127 %

全量条件查询速度提升

193 %

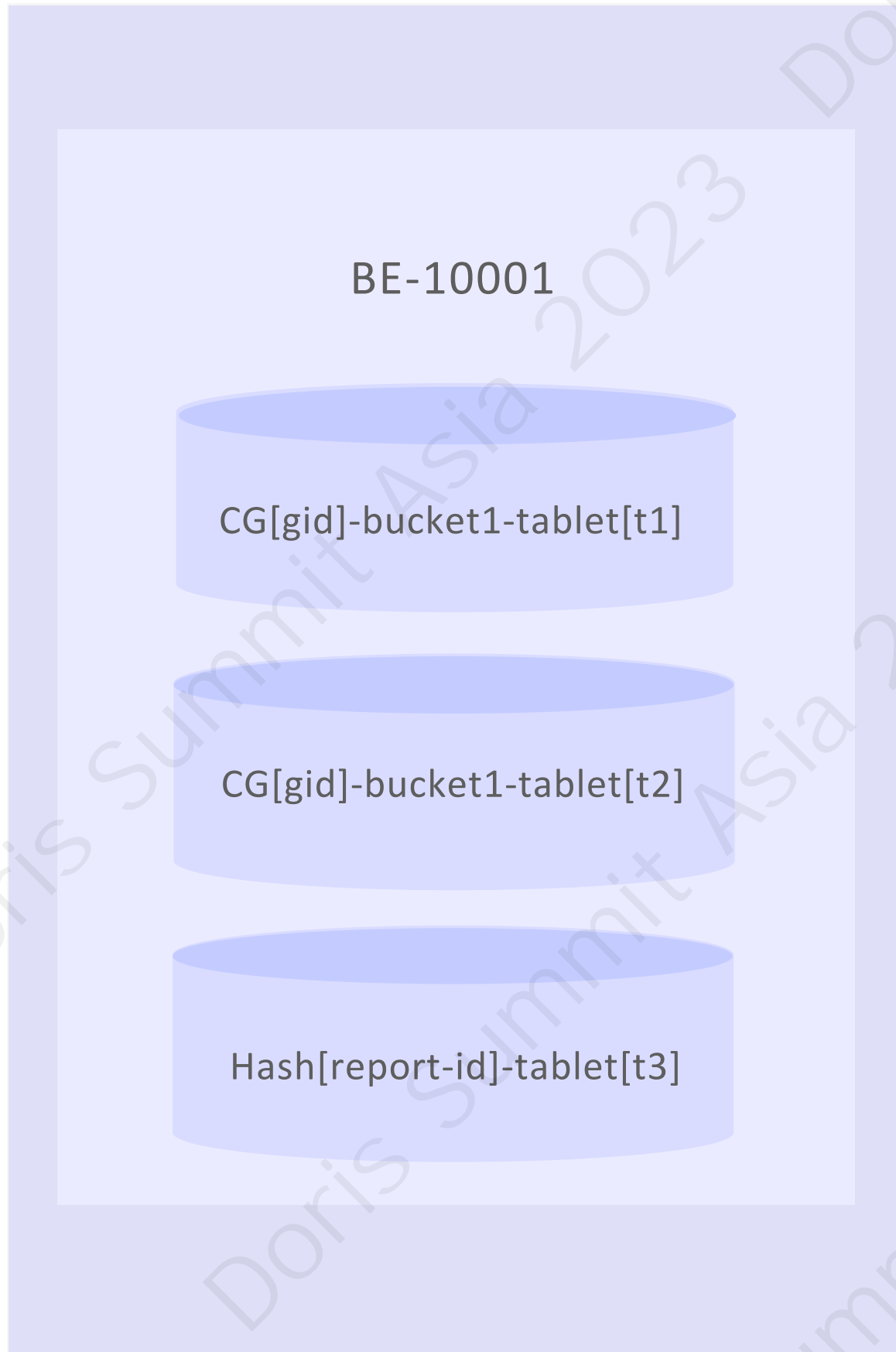
# Partial-update

## 合成企业基本信息宽表



```
@RequiredArgsConstructor
private static class Sink implements ForeachPartitionFunction<Row> {
    private final static Set<String> DIMENSION_KEYS = new HashSet<String>() {{
        add("...");
    }};
    private final Config config;
    @Override
    public void call(Iterator<Row> rowIterator) {
        ConfigUtils.setConfig(config);
        DorisTemplate dorisTemplate = new DorisTemplate("dorisSink");
        dorisTemplate.enableCache();
        // config `delete_on` and `seq_column` if is unique
        DorisSchema dorisSchema = DorisSchema.builder()
            .database("ads")
            .tableName("company_base_info").build();
        while (rowIterator.hasNext()) {
            String json = rowIterator.next().json();
            Map<String, Object> columnMap = JsonUtils.parseJsonObj(json);
            // filter needed dimension columns
            Map<String, Object> sinkMap = columnMap.entrySet().stream()
                .filter(entry -> DIMENSION_KEYS.contains(entry.getKey()))
                .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue));
            dorisTemplate.update(new DorisOneRow(dorisSchema, sinkMap));
        }
        dorisTemplate.flush();
    }
}
```

# Colocate-join & Bucket-join



```
CREATE TABLE `company_base_info` (  
  `company_id` bigint(20) NOT NULL,  
  `id` bigint(20) NOT NULL,  
  `company_name` string NOT NULL  
) ENGINE=OLAP  
UNIQUE KEY(`company_id`)  
COMMENT 'OLAP'  
DISTRIBUTED BY HASH(`company_id`)  
BUCKETS 64  
PROPERTIES (  
  "colocate_with" = "company"  
);
```

```
CREATE TABLE `annual_report` (  
  `company_id` bigint(20) NOT NULL,  
  `year` int(10) NOT NULL,  
  `id` bigint(20) NOT NULL  
) ENGINE=OLAP  
UNIQUE KEY(`company_id`,`year`)  
COMMENT 'OLAP'  
DISTRIBUTED BY HASH(`company_id`)  
BUCKETS 64  
PROPERTIES (  
  "colocate_with" = "company"  
);
```

```
CREATE TABLE `annual_report_shareholder_equity_info` (  
  `report_id` bigint(20) NOT NULL,  
  `id` bigint(20) NOT NULL,  
  `shareholder_equity_ratio` string NOT NULL  
) ENGINE=OLAP  
UNIQUE KEY(`report_id`,`id`)  
DISTRIBUTED BY HASH(`report_id`) BUCKETS 64;
```

Colocate-join速度提升

253 %

Bucket-join查询速度提升

77 %



# Light Schema Change



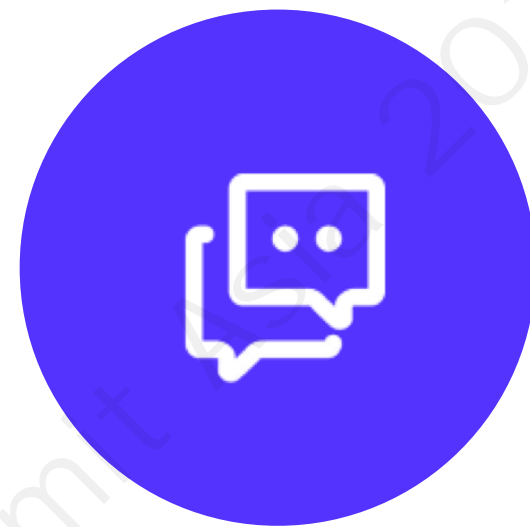
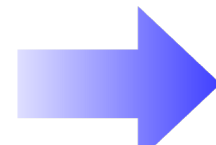
## Apollo Change

开发人员修改Apollo配置  
新增dim-count指标计算任务



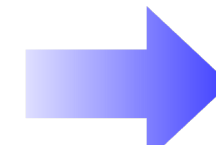
## Lock

Flink程序通过Listener  
监听到Apollo变更,  
请求Redis分布式锁



## Schema Change

第一个拿到锁的算子,  
判断success\_key, 发现不存在,  
执行schema change,  
结束后写success\_key

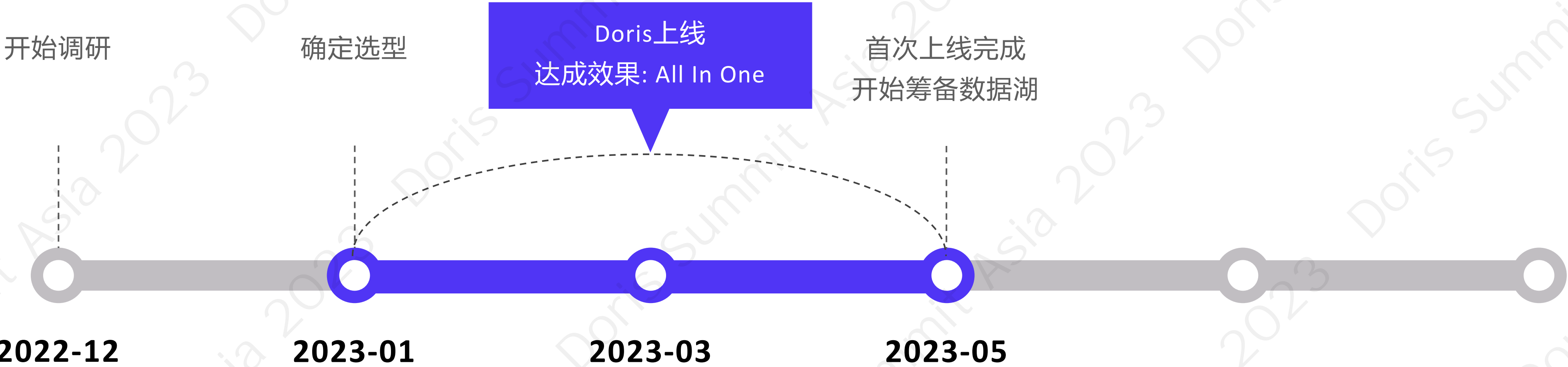


## Finish

其余算子依次验证success\_key后,  
继续执行dimcount计算

## 3 多源 Catalog 推动流批一体

# 第一阶段: All In One



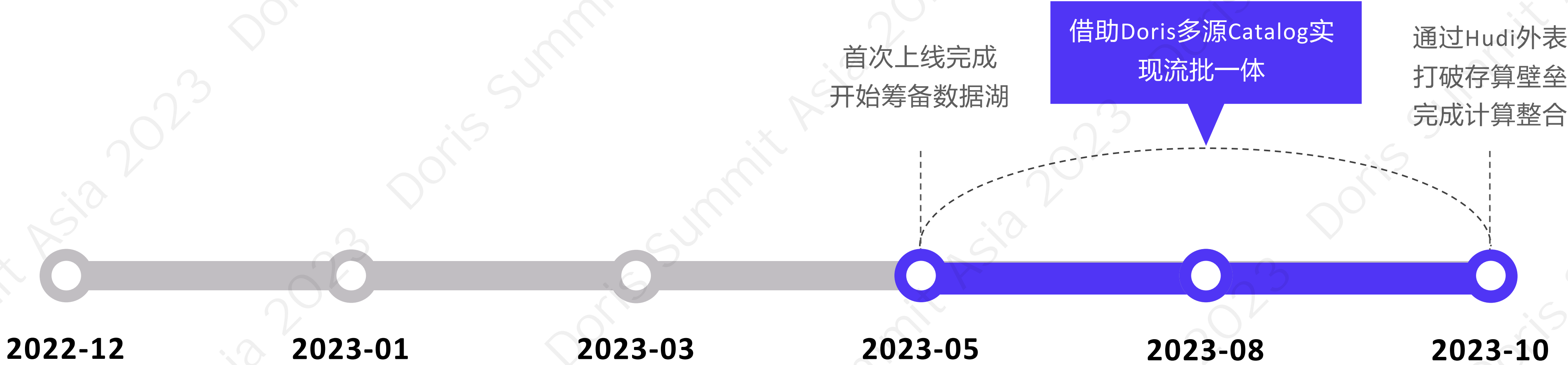
2023-03, Doris实现内部首次上线, 共计运行2个集群10台BE, 分别负责 数分团队商业化分析 与 数据平台部架构优化 ; 日更新数据量达10亿, 运行人群包、优惠券推送、充值订单分析, 以及数据交付等任务, 计算指标达500+

2023-05, 完成数分集群ETL任务的流式覆写, 提高集群稳定性; 同时实现离线定时调度任务迁移率25%, 内部集群BE数达到20, 使用Doris替代Hbase、GaussDB, 完成实时任务迁移率57%

未迁移重量离线计算任务计划后续借助Spark-Doris-Connector完成计算  
未迁移实时任务等待后续版本更强的高并发点查



## 第二阶段:流批一体



2023-05, All in Doris后, 依旧存在部分离线引擎才可以实现的超大规模数据计算, 同时离线ETL偶然的重新调度, 可能影响上线并发查询, 加上长期以来的Lambda架构, 历史遗留的维护成本愈发加大, 调研Doris的多源Catalog后, 希望在数据湖的加持下, C端等其它重要数据存在Doris, 其余数据使用存算分离的方式使用Doris加速查询

2023-08内部上线Hudi后, 截止至2023-10, 在多源Catalog的加持下,Doris专注应用层计算, 维护维度200+, 指标1200+, BE节点数已扩展至30+并持续增加中

## 4 架构演进的经验与总结

## -233(事务积压) & -235(版本过多)

Question	Answer
<ul style="list-style-type: none"><li>凌晨调度Broker-Load时, 因为调度系统任务挤压可能会导致同时并发多个 Load任务, be流量剧增, 出现写入毛刺, 导致事务堆积</li><li>新员工学习使用Broker-Load时, 同表短期导入失败5-7次后, 会偶现tablet异常</li></ul>	<ul style="list-style-type: none"><li>使用Stream-Load替代Broker-Load, 将流量分摊到全天</li><li>自定义写入器包装Stream-Load, 实现异步缓存、限流削峰等, 充分保证数据写入的稳定性</li><li>参数优化<ul style="list-style-type: none"><li>max_base_compaction_threads: 4-&gt;8</li><li>max_cumu_compaction_threads: 10-&gt;16</li><li>compaction_task_num_per_disk: 2-&gt;4-&gt;8</li><li>max_running_txn_num_per_db: 100-&gt;1000-&gt;2048</li><li>max_tablet_version_num: 500-&gt;1024</li></ul></li></ul>



# Fe 假死

## Question

- 通过Grafana监控, 经常发现Fe出现宕机现象, 通过“show frontends”, 并未发现异常

## Answer

- 早期搭建策略中, 因为Fe与Be端口并不冲突, 所以是混合部署, Be高强度计算时, 可能挤占同机器Fe的IO与内存
- Be机器的内存是128G, 使用32G的机器迁出Fe节点
- Stream-Load时直接指定Be的http端口, 不再请求Fe做转发, 降低Fe压力
- 定时调度“show processlist”命令, 探活的同时, 及时kill超时SQL或者超时连接

## 实时 Join 场景落地

### 多流join

- 一般业务形态强相关, 如下单数据与支付数据
- 数据会一直保存在内存中, 保留策略为设置DDL或者匹配即销毁
- 通常由Flink解决, 若是1:1 join, 可同时参考后续方案

### 多表 1:1 join

- 可以通过Doris Agg表模型的replace\_if\_not\_null功能实现prtial-update效果
- Agg表无法实现导入删除, 可以在结果表对每张上游表都预留一个is\_del字段, 根据业务确定最终圈选条件

### 多表 1:N:M join

- 每张表都可能错峰频繁变更, 单表自身凑不齐所有的unique-key-list, 依赖Flink计算, 需要对每张表都实现lookup-join逻辑
- 使用Doris MOW表,通过谓语句下推、命中索引、高性能join策略等, 实现现场关联

# 5 未来发展规划

## 未来规划与展望



### 线上升级2.0

更进一步的高并发点查



### 集群规模扩大至40+

基于多源Catalog  
实现存算分离  
更专注于计算性能



### 接入集群日志

多节点日志统一收集  
便于问题探查



### 自动化运维

找寻开源方案,  
解决集群升级、  
重启难题



### 提升数据质量

数据入口统一收束  
补齐数据质量监控



非常感谢Doris社区与SelectDB团队一直以来对我们的技术指导与支持,至此, 数据平台部已经使用Doris满足了实时与离线的统一写入与查询, 支持了BI分析、离线计算、C端高并发等场景, 为产品营销、客户运行、数据分析等场景提供数据洞察能力与价值挖掘能力, 未来,我们也会持续参与到ApacheDoris社区建设中,贡献更多在实时数仓的建设经验与实践应用





获取更多社区动态与最佳实践

### Apache Doris 官方平台:

- Apache Doris 官网: [doris.apache.org](https://doris.apache.org)
- Apache Doris GitHub: [github.com/apache/doris/](https://github.com/apache/doris/)

### 获取更多峰会资料:

- Doris Summit 峰会官网: [doris-summit.org.cn](https://doris-summit.org.cn)
- Doris Summit 峰会回放: <https://space.bilibili.com/1196172099/channel/collectiondetail?sid=1824324>