

ClickHouse在B站海量数据场景的落地实践

胡甫旺
哔哩哔哩OLAP平台



目录

- ❖ ClickHouse在B站
- ❖ 内核
- ❖ 日志
- ❖ 用户行为数据分析
- ❖ Future Work
- ❖ Q&A



ClickHouse在B站

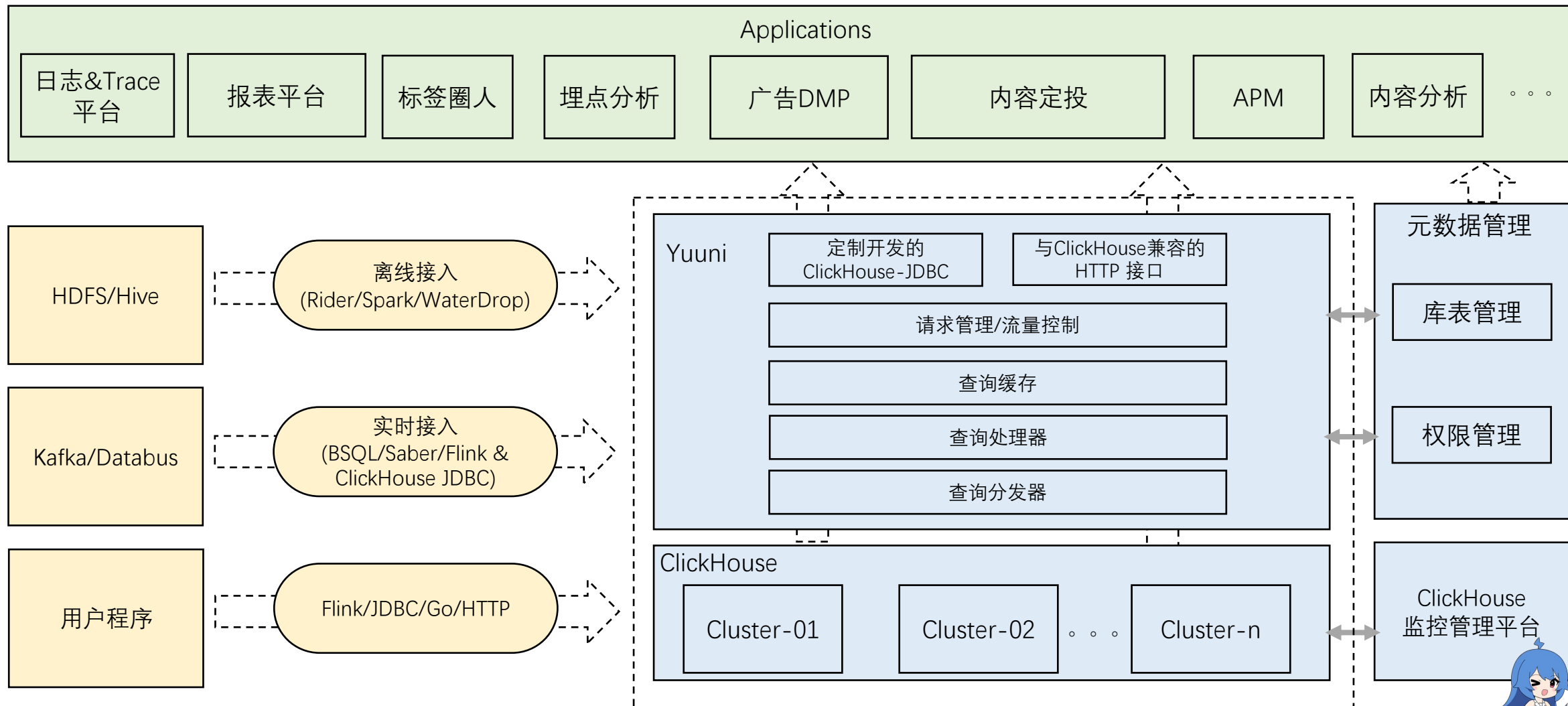


B站ClickHouse应用概况

- ❖ 近400个节点，30个集群
- ❖ 日均1.5+万亿条数据摄入
- ❖ 日均800+万次Select请求
- ❖ 应用场景包括（不限于）：
 - 日志&Trace分析
 - 用户行为分析（包括事件分析，漏斗分析，路径分析等）
 - 圈人定投
 - 广告DMP（包括统计分析，人群预估）
 - 电商交易分析
 - OGV内容分析
 - APM (Application Performance Management)



基于ClickHouse的交互式OLAP技术架构



ClickHouse as Service

❖ Berserker数据源管理:

- 建表
- 修改表元数据
- 表元数据管理

❖ Yuuni:

- 屏蔽集群信息
- 原生JDBC, HTTP接口
- 读写分离
- 动态查询缓存
- 流量控制

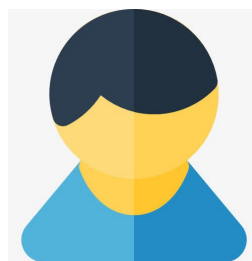
❖ 监控管理平台:

- 统计大盘
- 回归测试
- 接入评估
- 数据迁移
- 数据重平衡

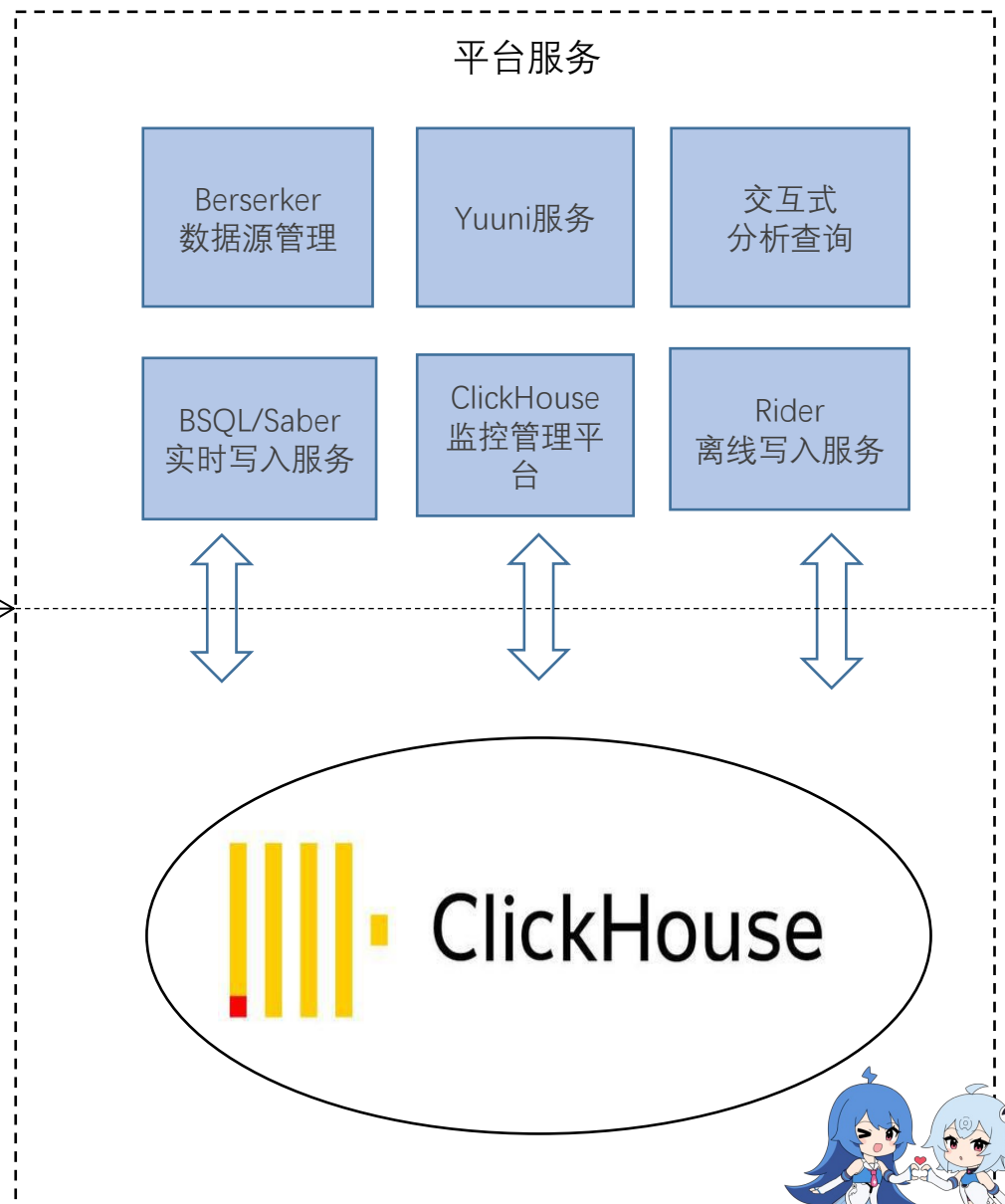
❖ 交互式分析查询: Superset提供即时查询能力

❖ 离线写入服务 (Rider)

❖ 实时写入服务 (BSQL/Saber)



用户

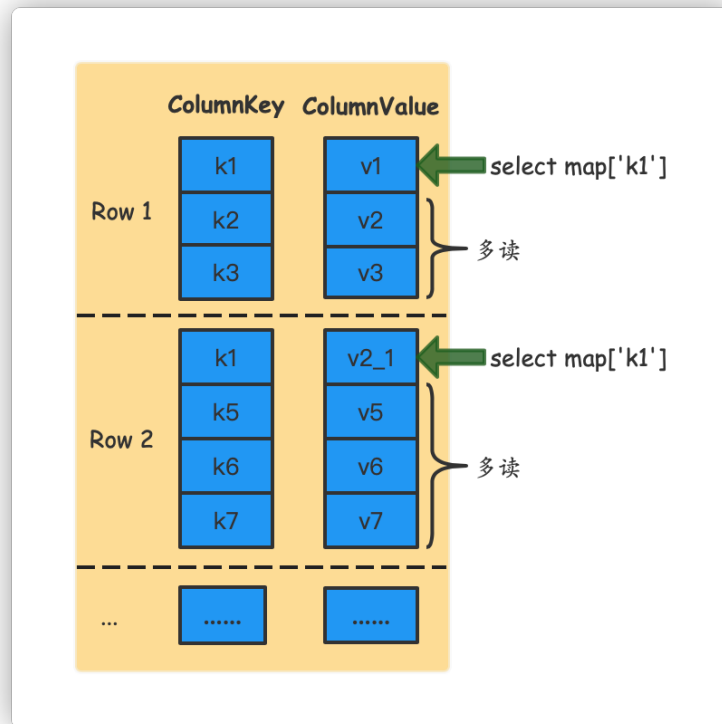
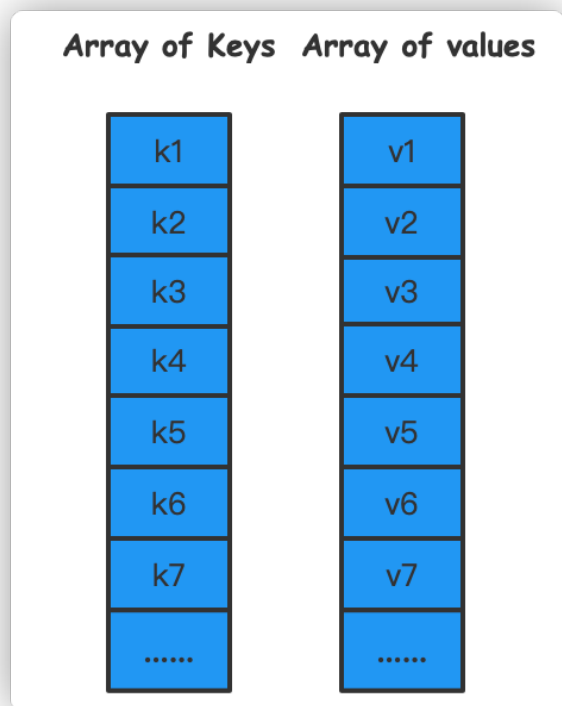


内核



Map隐式列

- ❖ 原生Map使用Array of Tuple实现
- ❖ 原生Map查询时需读取大量无效数据



Map隐式列

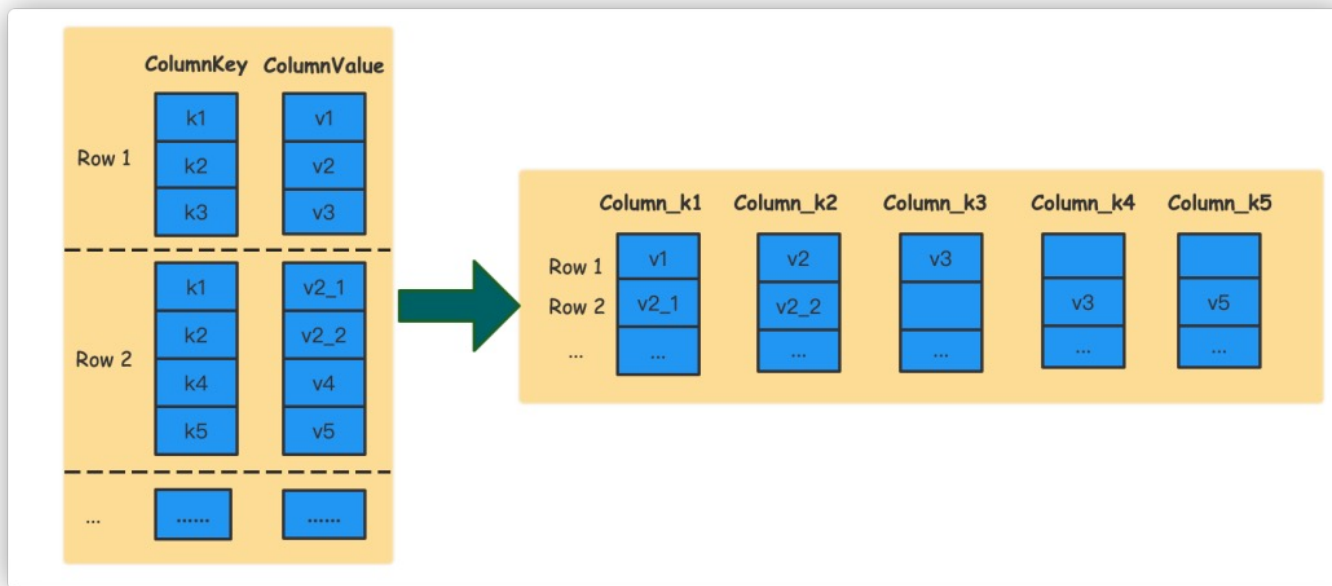
- ❖ Map隐式列将每个Key存储为独立列
- ❖ Map隐式列查询时只读取需要的隐式列

```
CREATE TABLE mapv2_table
(
  `id` UInt32,
  `map` MapV2(String, String)
)
ENGINE = MergeTree
ORDER BY id
```

Query id: f89ece2e-fd4a-421b-8f24-121e367c16f2

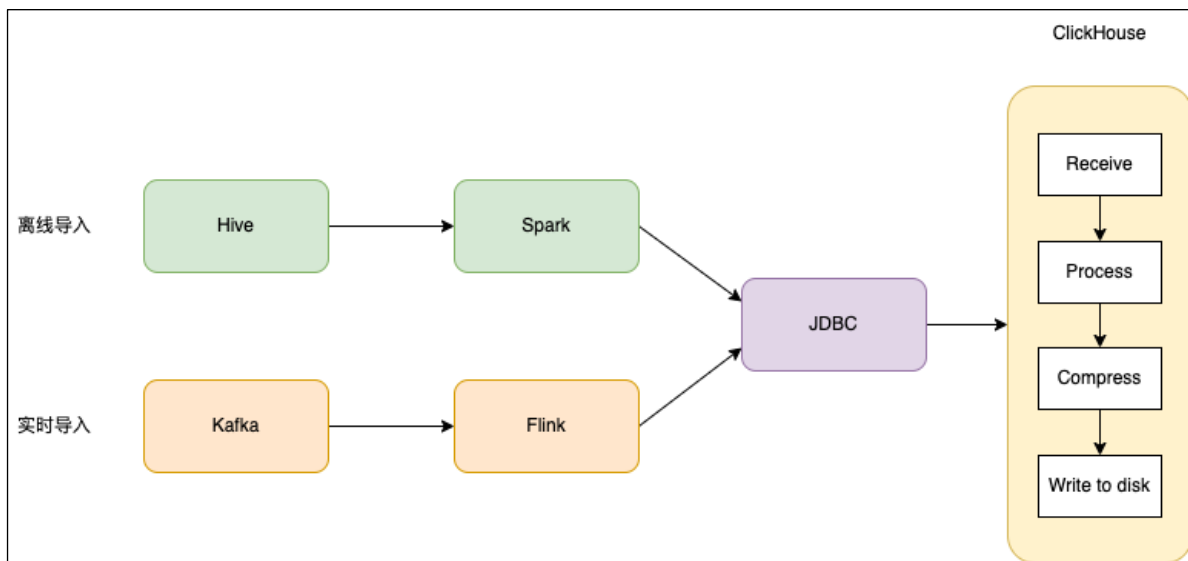
Ok.

0 rows in set. Elapsed: 0.003 sec.



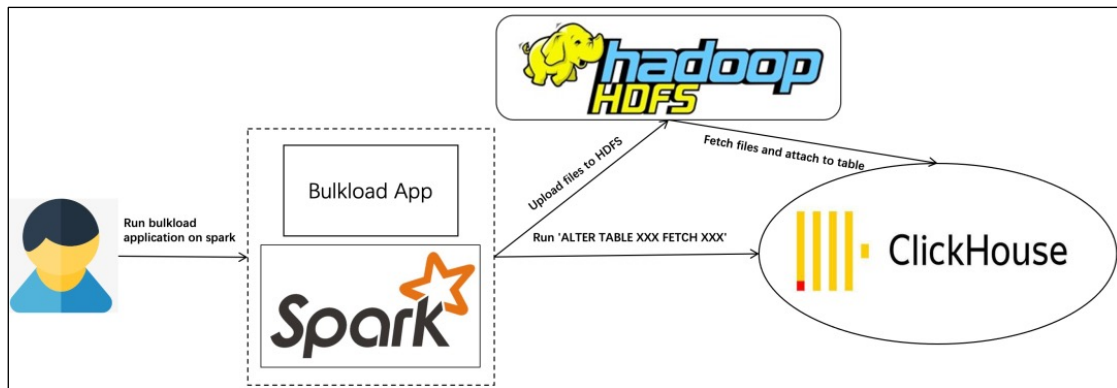
Bulkload

- ❖ 原生写入方式消耗ClickHouse Server资源，影响查询性能
- ❖ 实时写入任务长期占用资源，故障恢复的时间和运维成本较高
- ❖ 基于中间存储的Bulkload方案降低ClickHouse Server压力



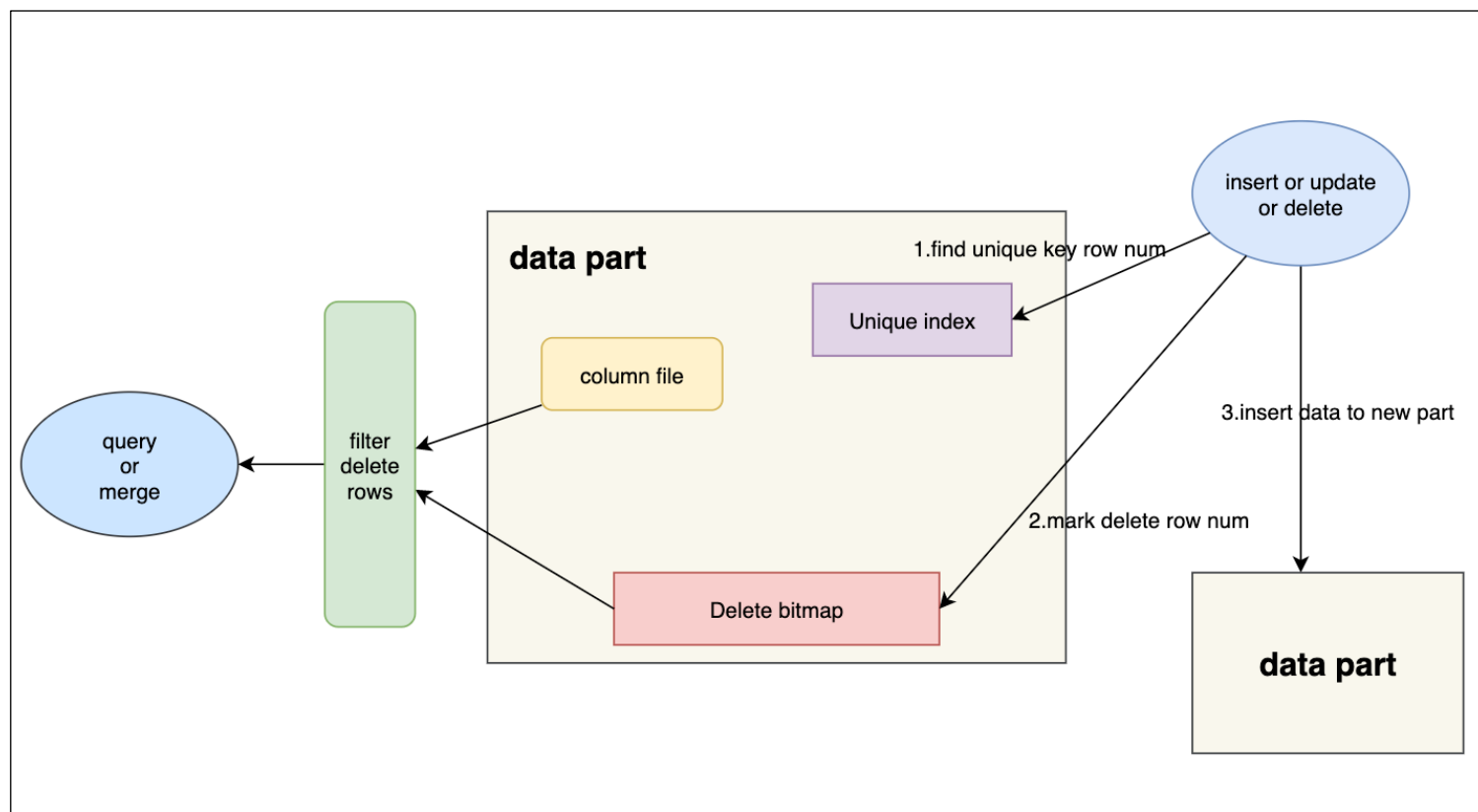
Bulkload

- ❖ 基于中间存储的Bulkload可以降低ClickHouse Server压力
- ❖ 基于中间存储的Bulkload受HDFS和网络稳定性影响，且传输成本较高
- ❖ 直达ClickHouse的Bulkload稳定性，性能都更佳



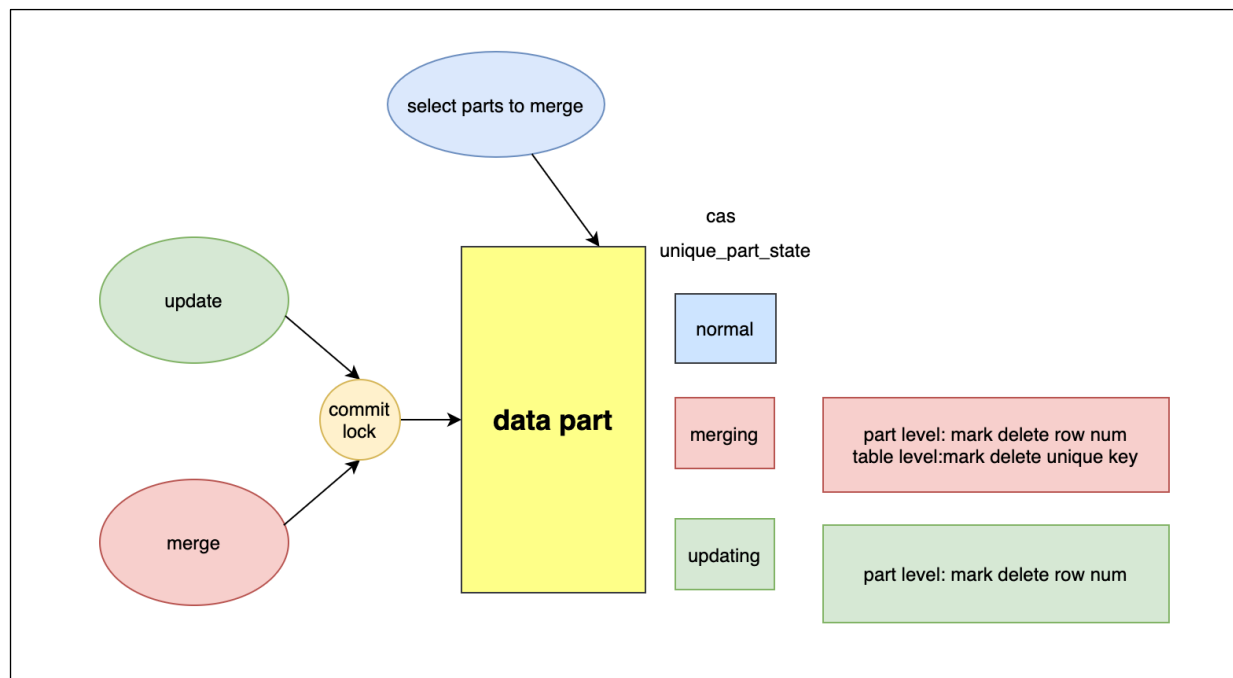
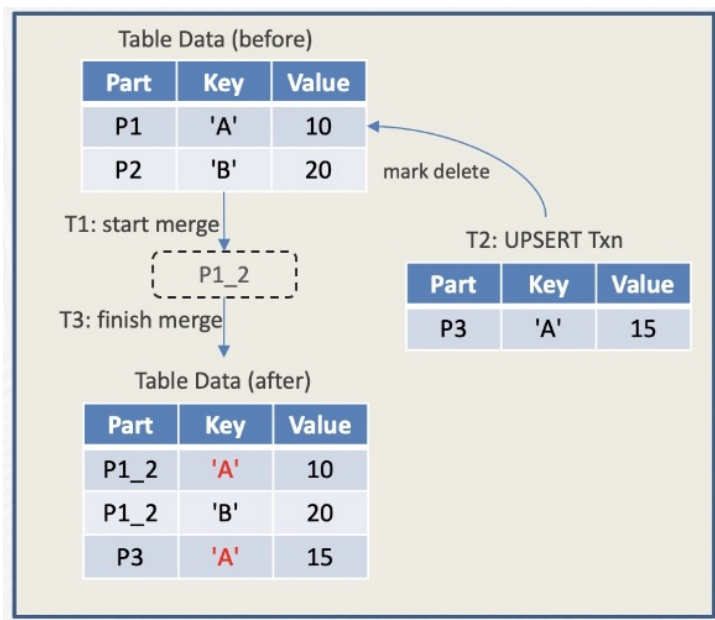
Unique Engine

- ❖ 目标：支持Upsert, Delete操作, 提升查询性能
- ❖ 设计：delete on insert



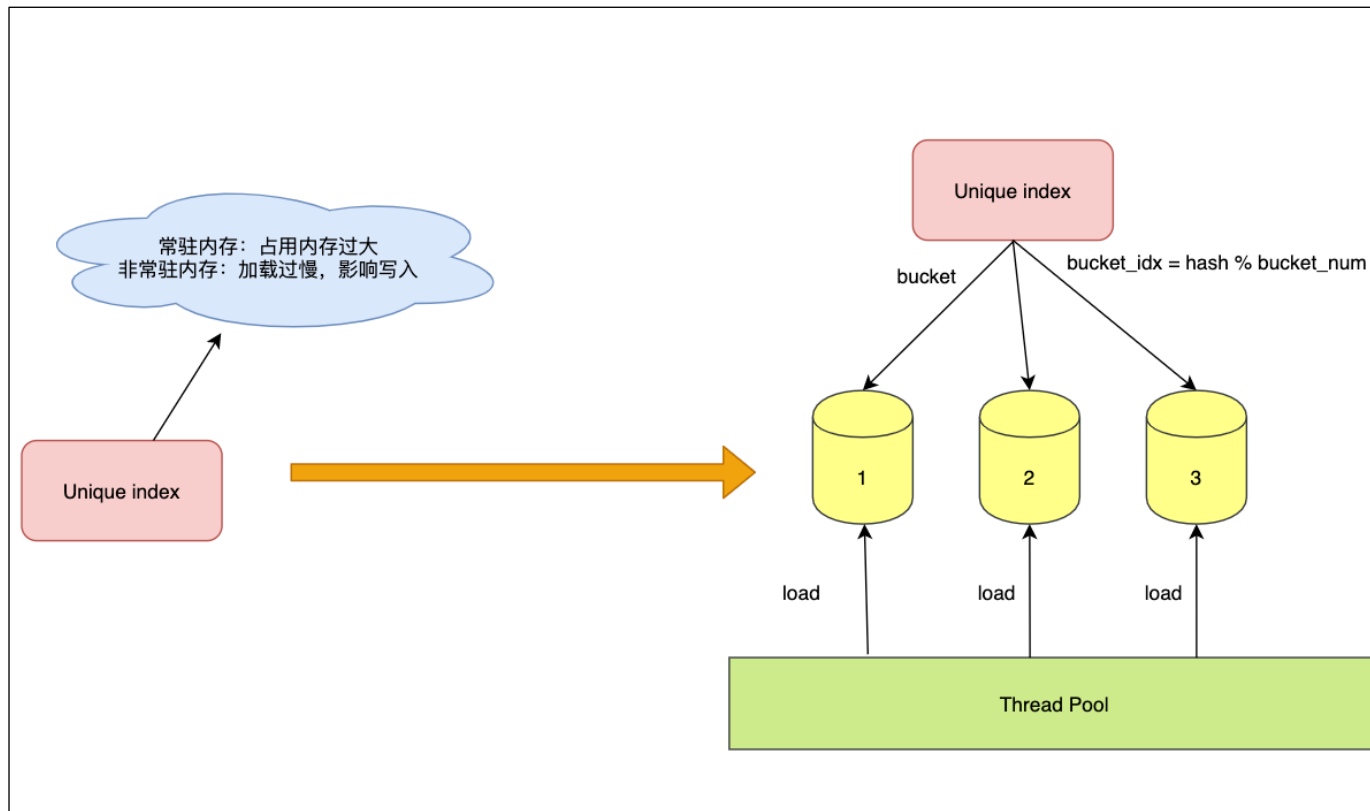
Unique Engine

- ❖ write-write冲突依靠table level lock控制
- ❖ write-merge冲突:



Unique Engine

- ❖ 常驻内存模式对内存消耗很大
- ❖ 非常驻内存模式index load过程慢
- ❖ 多并发加载优化索引加载速度:

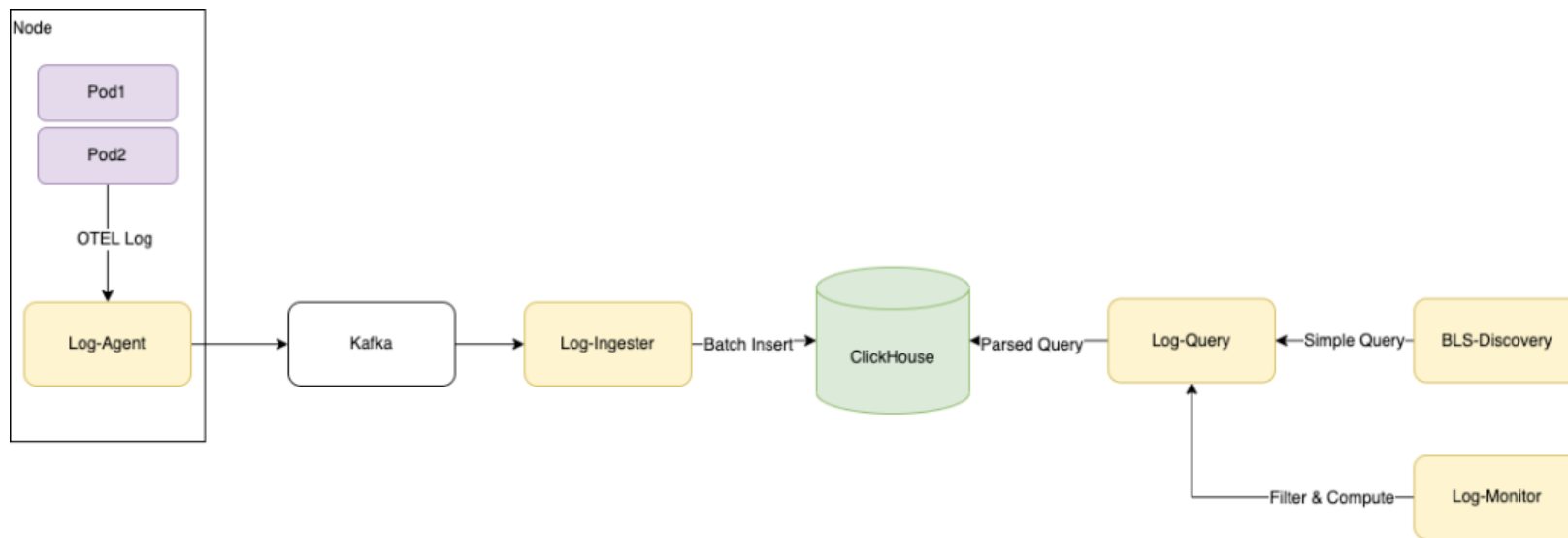


日志



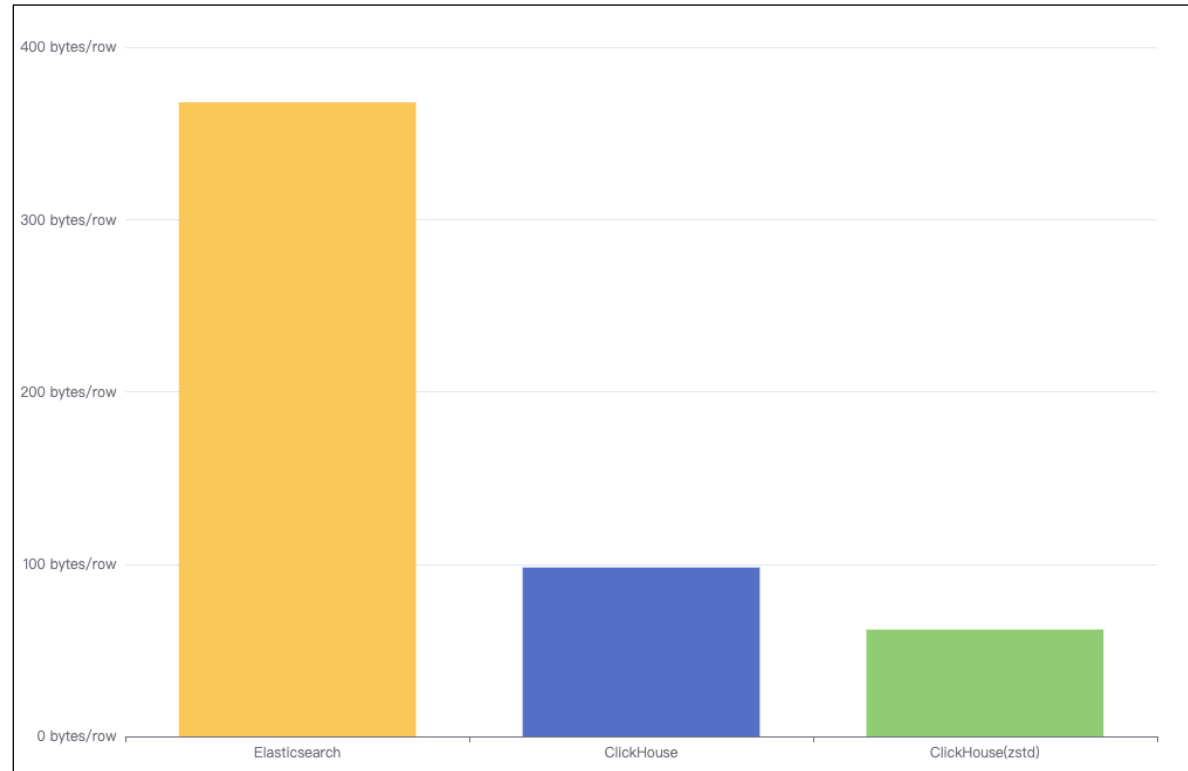
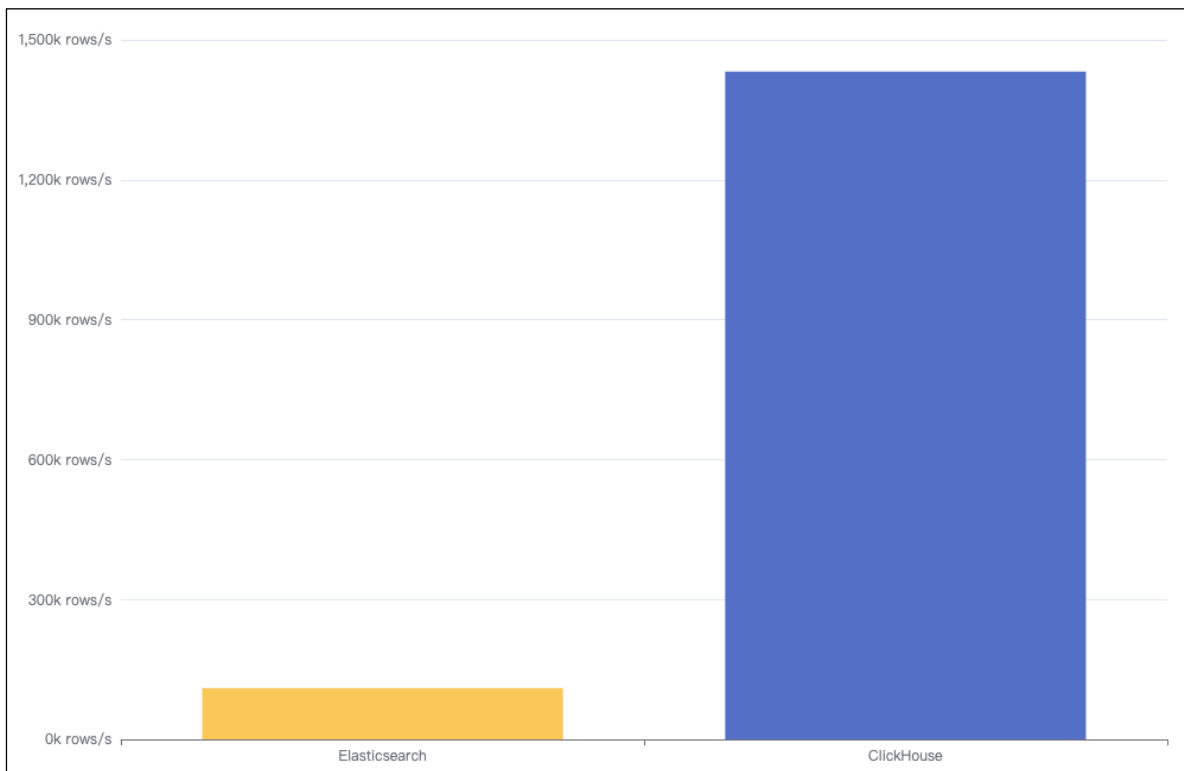
日志

- ❖ Elastic To ClickHouse迁移，降本增效
- ❖ OTEL标准化日志采集
- ❖ 统一scheme支持



日志

- ❖ ClickHouse较ES写入吞吐量提升近10倍
- ❖ ClickHouse存储成本为ES的1/3



日志

- ❖ ClickHouse中采用分表，统一schema的设计
- ❖ 日志查询采用类似ES语法，降低用户迁移成本

```
Create Table <log_app_name> ON CLUSTER ...
{
    _timestamp      Datetime64(3),
    `log.level`      String CODC(ZSTD(1)),
    `log.msg`        String CODC(ZSTD(1)),
    `log.trace_id`   String CODC(ZSTD(1)),
    ...
    string_map MapV2(String, Nullable (String))
                CODEC(ZSTD(1))
    number_map MapV2(String, Nullable (Float64))
                CODEC(ZSTD(1))
    bool_map MapV2(String, Nullable (UInt8))
}
ENGINE = ReplicatedMergeTree(...)
PARTITION BY toYYYYMMDD(_timestamp)
ORDER BY timestamp
TTL toDateTime(timestamp) + toIntervalDay(...),
    toDateTime(timestamp) + toIntervalDay(...) TO VOLUME 'cold_volume'
```

```
{
  "app_id": "demo-app",
  "query": "log.level = 'ERROR' AND user='fake-user'",
  "from": "2022-06-28 16:00:00",
  "to": "2022-06-28 17:00:00"
}
```

Log-Query

```
SELECT _timestamp, `log.level`, `log.msg`,
       `log.trace_id`, `log.span_id`,
       string_map{'user'} as user,
       string_map{'hostname'} as hostname,
       number_map{'duration'} as duration
FROM `demo_app_dist`
WHERE severity_text = 'ERROR'
AND _timestamp >= '2022-06-28 16:00:00'
AND _timestamp < '2022-06-28 17:00:00'
ORDER BY _timestamp DESC
LIMIT 500
```

```
{
  {
    "_timestamp": "2022-06-28 16:34:03.013",
    "_observed_timestamp": "2022-06-28 16:34:03.013",
    "log.level": "ERROR",
    "log.msg": "disconnection",
    "log.trace_id": "7c7a622eb165058103dd3fcb7b62baba",
    "log.span_id": "03dd3fcb7b62baba",
    "user": "fake-user",
    "host.name": "demo-app-0001"
  },
  {
    "_timestamp": "2022-06-28 16:07:03.012",
    "_observed_timestamp": "2022-06-28 16:07:03.012",
    "log.level": "ERROR",
    "log.msg": "error occur",
    "log.trace_id": "7c7a622eb165058103dd3fcb7b62baba",
    "log.span_id": "14sda25z3an2s81k",
    "user": "fake-user",
    "host.name": "demo-app-0001"
  },
  ....
}
```

_timestamp	log.level	log.msg	log.trace_id	log.span_id	user
2022-06-28 16:01:03.012	ERROR	disconnection	7c7a622eb165058103dd3fcb7b62baba	03dd3fcb7b62baba	fake-user
2022-06-28 16:02:03.012	ERROR	error occur	7c7a622eb165058103dd3fcb7b62baba	14sda25z3an2s81k	fake-user



用户行为数据分析



概述

- ❖ 基于ClickHouse构建B站用户行为数据分析产品：北极星
- ❖ 行为数据分析平台主要以下功能模块：



事件分析

查看埋点基础PV、UV、点击率等



漏斗分析

多个步骤间转化效果



页面分析

批量查找一批埋点基础数据



路径分析

访问前向来源与后续行为分布



留存分析

用户在产品内留存情况



SQL自定义查询

使用 SQL 分析埋点数据



单用户行为细查

查看埋点基础PV、UV、点击率等



用户分群

根据用户行为和属性创建人群并分析



事件分析

- ❖ 海量埋点事件数据，日增数据千亿级。
- ❖ 用户行为事件的多维度分析场景。
- ❖ 事件包含公共属性和私有属性，均可作过滤和聚合维度。
- ❖ 不同事件有不同的私有属性字段。
- ❖ 动态选择的过滤维度和聚合维度。
- ❖ 交互式分析延迟要求 (5秒内)。

埋点事件选择 ?

A [模糊输入] 的 总次数

添加别名 添加事件 复合指标 ?

+ 属性筛选 + 用户筛选 + 用户人群

且

房间号 (room_id) 等于= 搜索或输入自定义值 批量录入 x

手机品牌 (brand) 等于= Apple 批量录入 x

性别_buvid 等于= 男 批量录入 x

属于 [模糊输入] x

按分组 总体 搜索 合并去重 ?

全局筛选 + 公共属性筛选

北极星人群

用户画像人群

AB实验人群

[模糊输入]

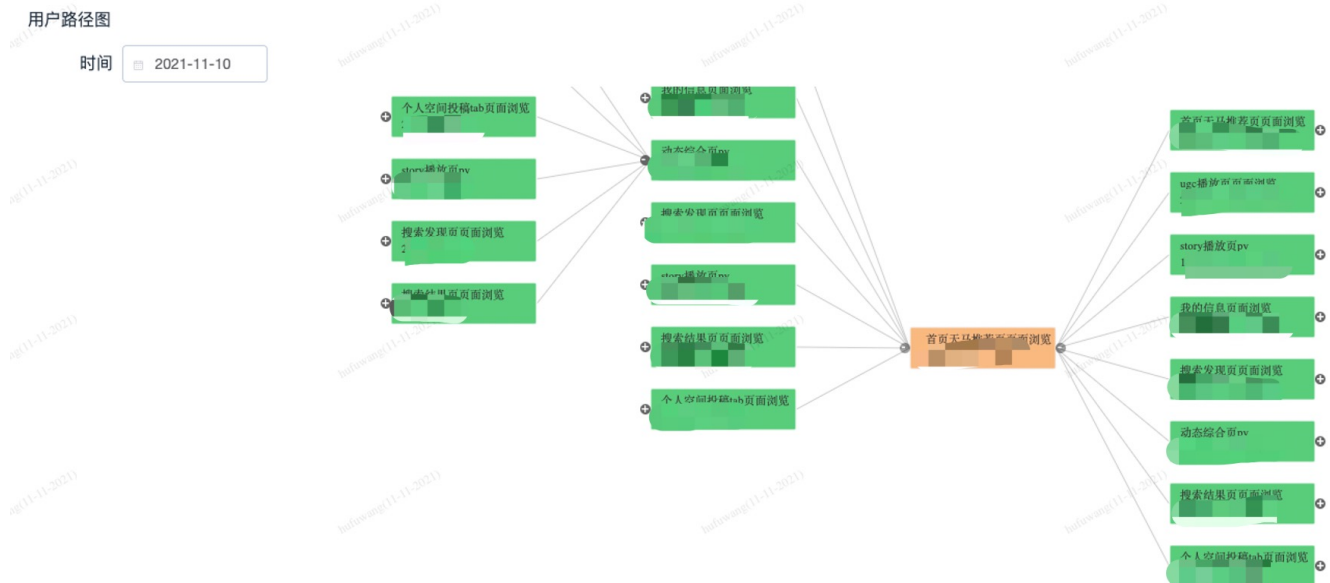
[模糊输入]

查询 保存



路径分析

- ❖ 选定中心事件。
- ❖ 按时间窗口确定上下游事件。
- ❖ 离线Spark与计算出事件路径及相关用户id的RBM。
- ❖ 离线计算结果导入ClickHouse做交互式路径分析。



漏斗分析

- ❖ 预定义事件漏斗
- ❖ 支持各个事件单独设置过滤条件
- ❖ 查询时间跨度最大一个月
- ❖ 数据按user id做Sharding，查询下推

```
select level, count(distinct buvid)
from (
  SELECT
    buvid,
    windowFunnel(3600)(time_iso, event_id = 1, event_id = 2, event_id = 3, event_id = 4) AS level
  from event_analysis_table_dst
  where log_date= '20201201' and event_id in (1, 2, 3, 4)
  GROUP BY buvid
  settings distributed_group_by_no_merge=1
) group by level
```

```
select level, sum(cnt) as cnt
from cluster( 'cluster_name',
view(
  select level, count(distinct buvid) as cnt
  from (
    SELECT
      buvid,
      windowFunnel(3600)(time_iso, event_id = 1, event_id = 2, event_id = 3, event_id = 4) AS level
  from event_analysis_table_local
  where log_date = 20201201' and event_id in (1, 2, 3, 4)
  GROUP BY buvid
  ) group by level)
) group by level
```



Future Work



Future Work

- ❖ ClickHouse集群容器化，提升物理集群资源使用率
- ❖ ClickHouse倒排索引调研与改造，提升日志检索性能
- ❖ 丰富ClickHouse编码类型，拓展zorder应用场景，提升圈选计算性能
- ❖ ClickHouse存算分离探索，降低集群扩容成本



Q&A

