基于ClickHouse的海量数据交互式分析场景实践

胡甫旺
哔哩哔哩OLAP平台

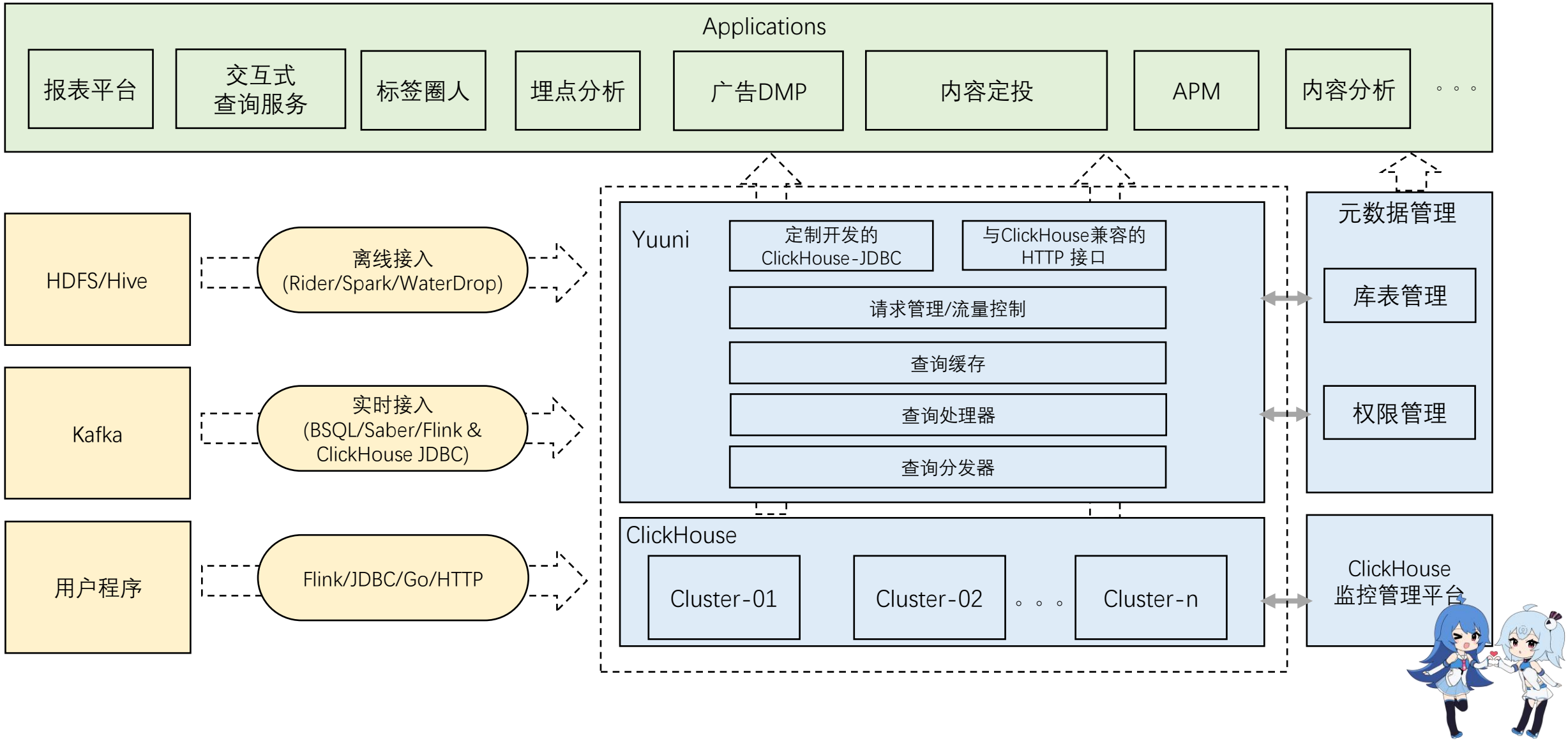
目录

ClickHouse在B站

B站ClickHouse应用概况

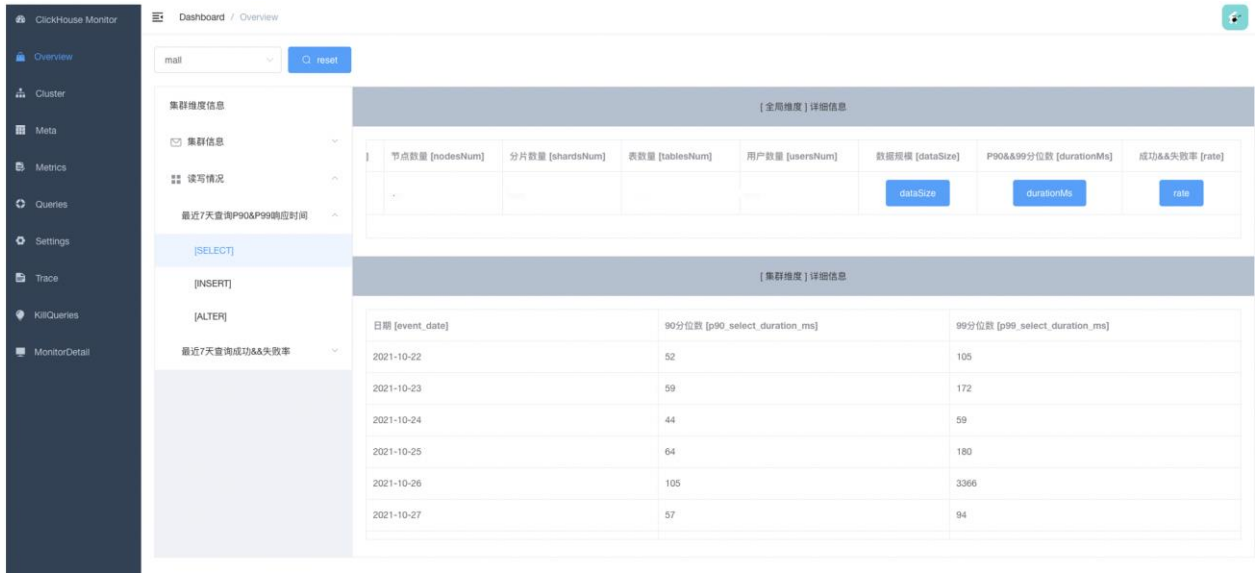
- 近200个节点，20+个集群
- 日均4000+亿条数据摄入
- 日均450+万次Select请求
- 日均2000+万次Insert请求
- 应用场景包括（不限于）：
  - 用户画像分析
  - 圈人定投
  - 广告DMP（包括统计分析，人群预估）
  - 电商交易分析
  - OGV内容分析
  - 日志/Trace分析
  - APM (Application Performance Management)

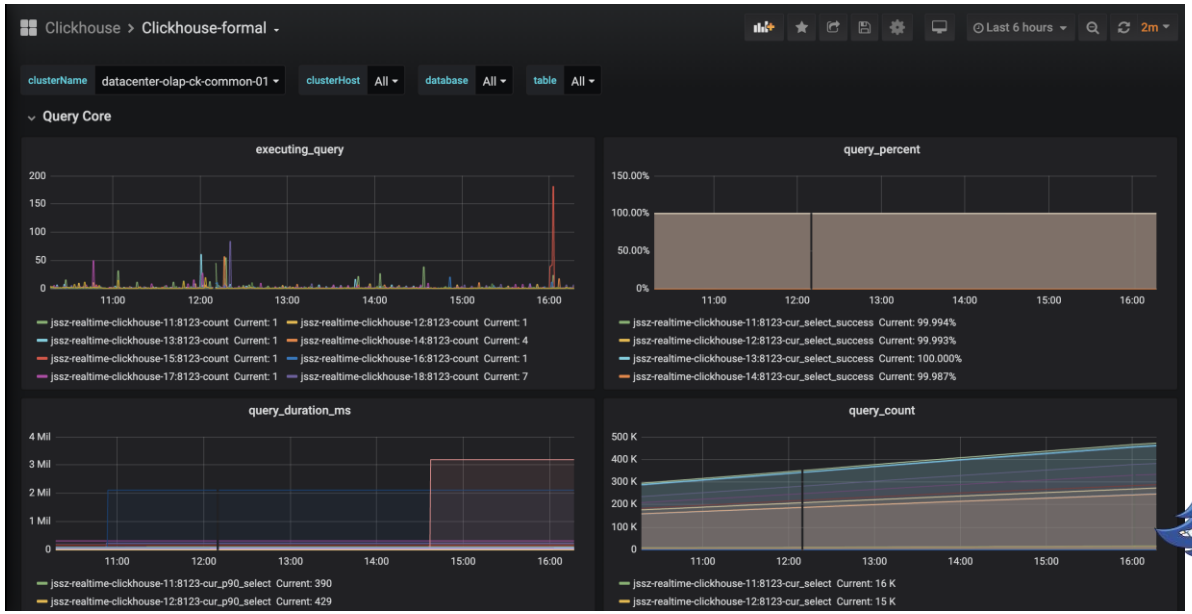
基于ClickHouse的交互式OLAP技术架构



ClickHouse集群规范化管理

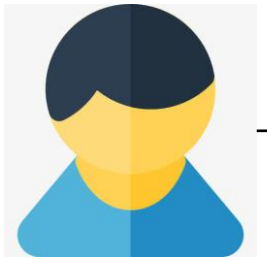
- 集群信息规范化：
  - 服务树规范
  - 机器命名规范
  - 监控指标统一规范
  - 库表命名规范
- 业务接入规范化：
  - 建表
  - 数据导入
  - 测试流程
  - 风险评估
- 集群操作规范和服务化：
  - 新建集群
  - 集群扩容
  - 数据迁移
  - 故障修复



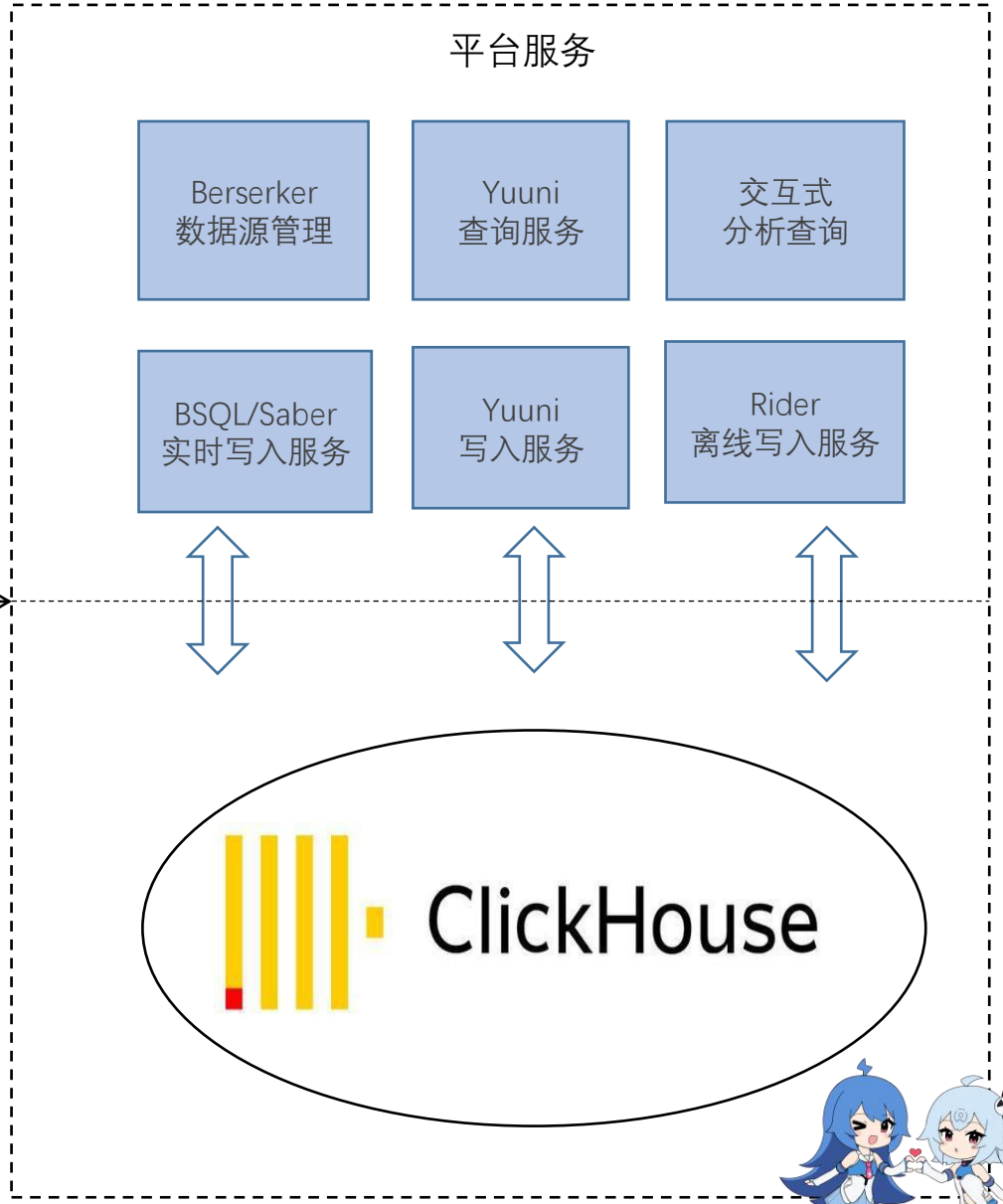


ClickHouse as Service

- 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
- 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



场景分析

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)。

事件分析

SQL-V1：

```
SELECT log_date, brand, indicator
FROM (
    SELECT log_date, brand, SUM(pv) AS indicator
    FROM user_event_table
    WHERE log_date BETWEEN '20211103' AND '20211103'
        AND event_id = 'event_1'
        AND app_id = 1
        AND extended_field_values[indexOf(extended_field_keys, 'goto')]='new_tunnel'
        AND extended_field_values[indexOf(extended_field_keys, 'sub_goto')]='activity'
    GROUP BY log_date, brand

    UNION ALL

    SELECT '' AS log_date, brand, SUM(pv) AS indicator
    FROM  user_event_table
    WHERE log_date BETWEEN '20211103' AND '20211103'
        AND event_id = 'event_1'
        AND app_id = 1
        AND extended_field_values[indexOf(extended_field_keys, 'goto')]='new_tunnel'
        AND extended_field_values[indexOf(extended_field_keys, 'sub_goto')]='activity'
    GROUP BY brand
) AA
ORDER BY indicator DESC
LIMIT 500;
```

SQL-V2：

```
SELECT log_date, brand, SUM(pv) AS indicator
FROM user_event_table
WHERE log_date BETWEEN '20211103' AND '20211103'
    AND event_id = 'event_1'
    AND app_id = 1
    AND extended_props['goto']='new_tunnel'
    AND extended_props['sub_goto']='activity'
GROUP BY GROUPING SETS ((log_date, brand), (brand))
ORDER BY SUM(pv) DESC
LIMIT 500;
```

漏斗分析

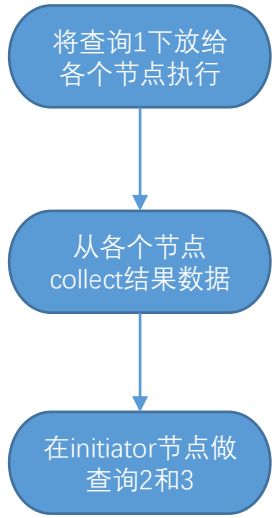
- 预定义事件漏斗。
- 支持各个事件单独设置过滤条件。
- 查询时间跨度最大一个月。
- 数据按user id做Sharding，查询下推。

漏斗分析

原始SQL:

```
SELECT level, uniq(buvid)   ③
from (
SELECT   ②
    buvid,
    windowFunnel(86400)(time_iso, event_id = 1, event_id = 2, event_id = 3, event_id = 4) AS level
FROM
(          ①
    SELECT
        time_iso,
        event_id,
        buvid
    FROM event_analysis_table_dst
    where log_date='20201201'and event_id in (1, 2, 3, 4)
) GROUP BY buvid
) GROUP BY level
```

执行步骤:

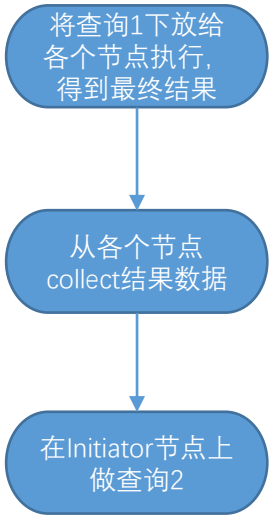


漏斗分析

优化SQL-V1 （数据按buvid做sharding）：

```
select level, count(distinct buvid)  ②
from (
    SELECT  ①
    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
```

执行步骤:



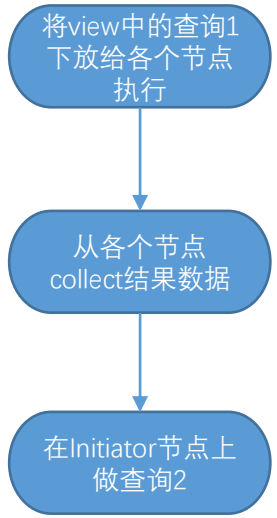
优化结果：较原始SQL性能提升**5+**倍

漏斗分析

优化SQL-V2（数据按buvid做sharding）：

```
select level, sum(cnt) as cnt                                                    ②
from cluster('cluster_name',
view(
 select level, count(distinct buvid) as cnt                                      ①
  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
```

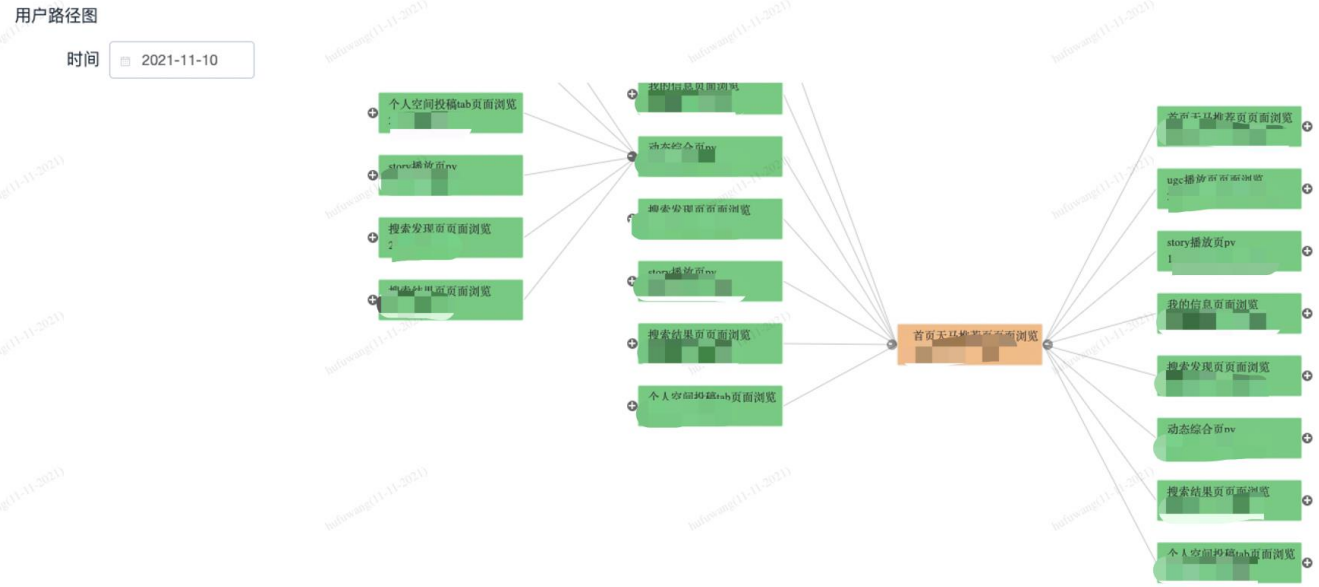
执行步骤:



优化结果：较优化SQL-V1性能提升**30+%**

路径分析

- 选定中心事件。
- 按事件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。



路径分析

```
CREATE TABLE polaris.polaris_dws_flow_app_path_eventid_i_a_d_v3_local
(
    `event_id` String,
    `event_ids` Array(String),
    `path_direction` Int16,
    `buvids_rbm_string` String,
                    ...
    `path_lenth` Int32,
    `buvids_roaring_bitmap` AggregateFunction(groupBitmap, UInt64)
MATERIALIZED base64Decode(buvids_rbm_string),
    INDEX platform_idx platform TYPE minmax GRANULARITY 3
);
```

```
SELECT log_date, path, path_direction, bitmapCardinality(groupBitmapOrState(buvids_roaring_bitmap)) AS uv
FROM
(
    SELECT log_date, path, path_direction, buvids_roaring_bitmap
    FROM cluster('polaris_cluster_replica', view(
        SELECT log_date,  path_direction,  [event_ids[1]] AS path,
                groupBitmapOrState(buvids_roaring_bitmap) AS buvids_roaring_bitmap
        FROM polaris.polaris_dws_flow_app_path_eventid_i_a_d_v3_local
        WHERE (log_date = '20211105') AND (event_id = 'mall.magic-item-view.0.0.pv')
                AND (app_id = 1) AND (path_lenth = 5)
        GROUP BY log_date, path, path_direction
    ))
)
GROUP BY log_date, path, path_direction
UNION ALL
。。。
```

引擎增强

ClickHouse引擎内核增强

Map数据类型

- 适用于动态列场景，较Array易用性更强。
- 添加多种类型索引支持，加速map取数。
  - bloom_filter
  - tokenbf_v1
  - ngrambf_v1

```
CREATE TABLE polaris.polaris_event_analysis_table_v2_local
(
    `buvid` String CODEC(ZSTD(15)),
                ...
    `extended_field_keys` Array(String),
    `extended_field_values` Array(String),
    ...
)
...
```

支持Map类型后

```
CREATE TABLE polaris.polaris_event_analysis_table_v2_local
(
    `buvid` String CODEC(ZSTD(15)),
    ...
    `extended_fields` Map(String, String),
    INDEX idx_extended_fields TYPE tokenbf_v1(256, 3, 0) GRANULARITY 2
    ...
)
...
```

```
SELECT
    buvid, XXX
FROM polaris.polaris_event_analysis_table_v2_local
WHERE extended_field_values[indexOf(extended_field_keys,'goto')] = 'banner'
```

支持Map类型后

```
SELECT
    buvid, XXX
FROM polaris.polaris_event_analysis_table_v2_local
WHERE extended_fields['goto'] = 'banner'
```

ClickHouse引擎内核增强

Map数据类型

- 添加自定义map函数，支持map数据过滤。
  - mapContainsKeyLike
  - mapExtractKeyLike
- 添加map函数对索引的支持，提升map函数性能。

```
SELECT
    buvid,
    mapExtractKeyLike(extended_fields, '%game%')
FROM polaris.polaris_event_analysis_table_v2_local
WHERE mapContainsKeyLike(extended_fields, '%game%')
```

ClickHouse引擎内核增强

Grouping Sets

- 支持多个维度组合做聚合。
- 简化用户SQL，多个查询合并为单个查询。
- 大幅减少磁盘读取开销。

```
SELECT
        log_date AS logDate,
        'all' as event_id,
        CAST(uniq(buvid_l), 'Float64') AS indicator
FROM polaris.polaris_event_analysis_table_local
GROUP BY log_date

UNION ALL

SELECT
        'all' AS logDate,
        event_id,
        CAST(uniq(buvid_l), 'Float64') AS indicator
FROM polaris.polaris_event_analysis_table_local
GROUP BY event_id
```

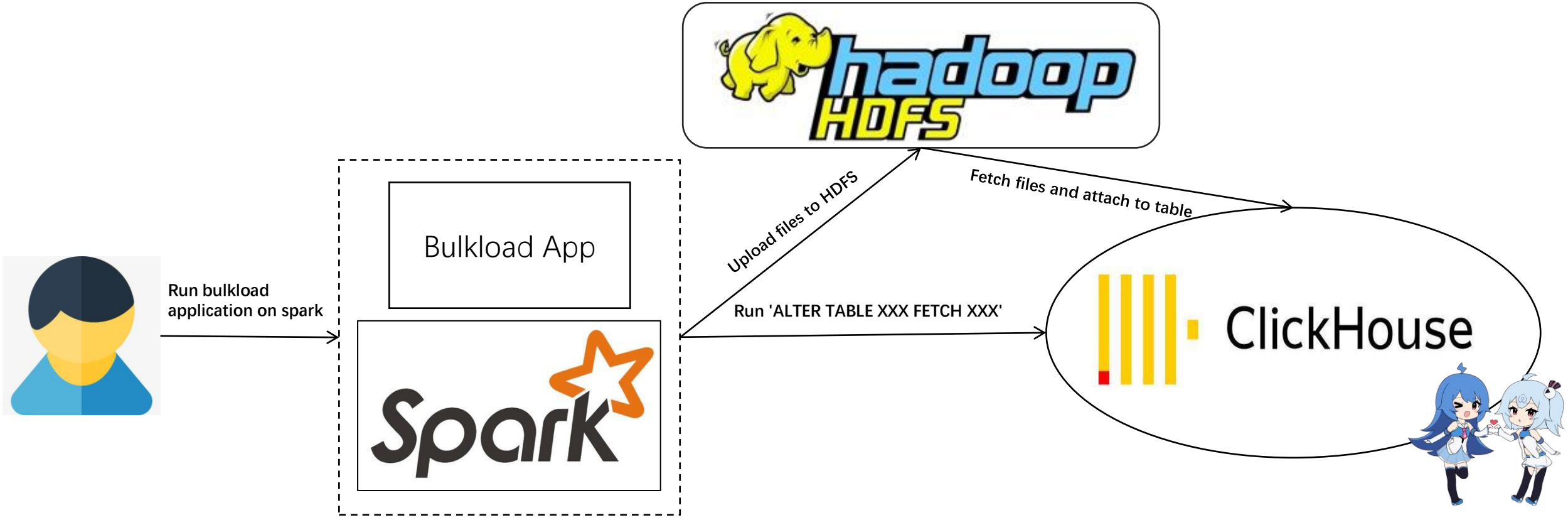
支持Grouping Sets后

```
SELECT
        if(log_date='', 'all', log_date) AS log_date,
        if(event_id='', 'all', log_date) AS event_id,
        CAST(uniq(buvid_l), 'Float64') AS indicator
FROM polaris.polaris_event_analysis_table_local
GROUP BY GROUPING SETS ((log_date), (event_id))
```

ClickHouse引擎内核增强

ClickHouse Bulkload

- ClickHouse数据文件生成过程的外置到Spark application。
- ClickHouse支持ALTER TABLE … FETCH 'hdfs://xxx'
- 数据导入过程几乎不占用ClickHouse的CPU，内存资源。



ClickHouse引擎内核增强

Z-ORDER

- 支持多字段构造zvalue做主键索引。
- 配合跳数索引加速多场景单字段过滤查询。
- 加速多维度范围查询场景。
- 快速跳过zcurve seam。

ClickHouse引擎内核增强

Z-ORDER

```
CREATE TABLE user_profiling
(
  buvid String,
  game_watch_duration_week UInt32,
  science_watch_duration_week UInt32,
  ...
  log_date String
)
PARTITION BY log_date
ORDER BY zCurve(game_watch_duration_week, science_watch_duration_week)
...
```

```
SELECT countDistinct(buvid) as uv
FROM user_profiling
WHERE game_watch_duration_week >= 3600
and game_watch_duration_week <= 10*3600
and science_watch_duration_week > 5*3600
and science_watch_duration_week <= 10*3600
```

```
CREATE TABLE user_profiling
(
  buvid String,
  game_watch_duration_week UInt32,
  science_watch_duration_week UInt32,
  ...
  log_date String,
  INDEX idx_1 game_watch_duration_week TYPE minmax GRANULARITY 2,
  INDEX idx_2 science_watch_duration_week TYPE minmax GRANULARITY 2
)
PARTITION BY log_date
ORDER BY zCurve(game_watch_duration_week, science_watch_duration_week)
...
```

```
SELECT countDistinct(buvid) as uv
FROM user_profiling
WHERE game_watch_duration_week >= 3600
and game_watch_duration_week <= 10*3600
```

Future Work

Future Work

- 实现ClickHouse离线和实时写入的原子性
- 集群虚拟化：ClickHouse on k8s
- Map类型进一步改造：用隐式列实现Map类型
- 存算分离方案实现
- 更多索引类型支持：如支持Bit-sliced index

Q&A