DTCC

数，造，未，来

第十二届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2021

2021 年 10 月 18 日 - 20 日 | 北京国际会议中心

Spark
Hadoop
OLAP
HANA
MPP
Oracle
SQL

IT168.com ChinaUnix.net ITPUB

滴普基于ClickHouse的实时分析引擎应用实践

滴普科技 陈峰

ClickHouse简介

特点

- 列式OLAP数仓
- 查询速度快
- 单机性能高
- 支持SQL

不擅长的场景

- 不支持事务
- 不适合点查
- 仅能批量删除或修改数据

ClickHouse在滴普的相关应用

DataSense

基于ClickHouse+AI构建的ABI应用

DBMS4CK

ClickHouse安装、升级、管理工具

DCT

为ClickHouse优化的数据传输工具

在现实应用中的一些痛点

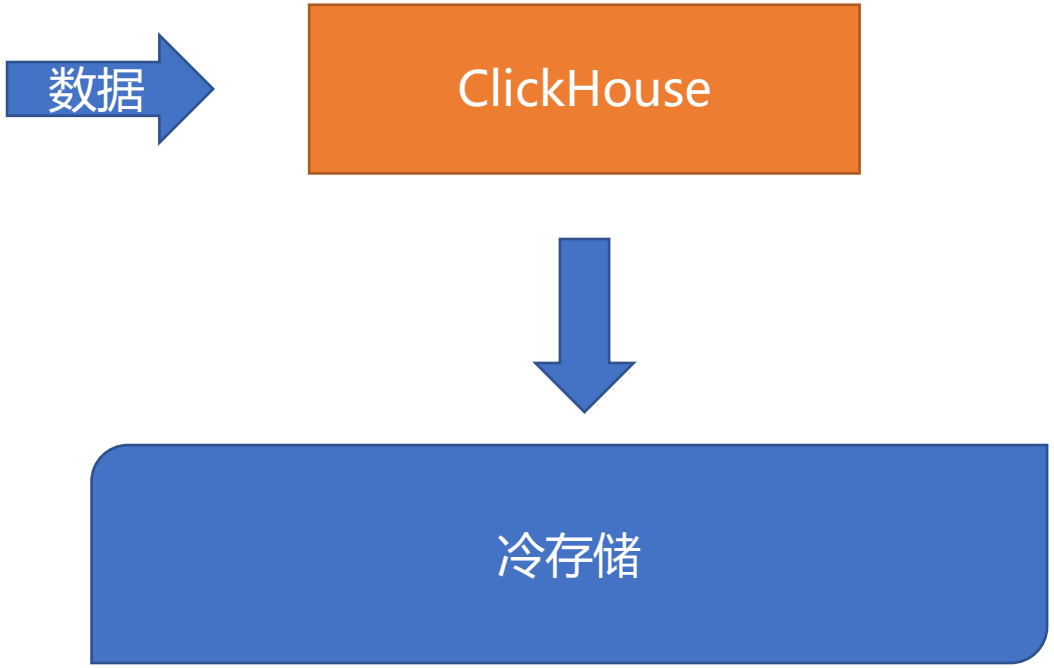
- 企业已经有了大数据集群，如何融合现有hadoop生态体系
- ClickHouse存在一些限制，有些场景无法单独使用

- 解决方案：冷热数据分离存储

ClickHouse 冷热数据分离存储

实现冷热分离

热—>冷



- 数据进入ClickHouse即为热数据
- 冷却后进入冷存储
- 实现简单、借助ClickHouse的TTL机制+存储可以实现
- 冷数据和热数据都可以被ClickHouse处理

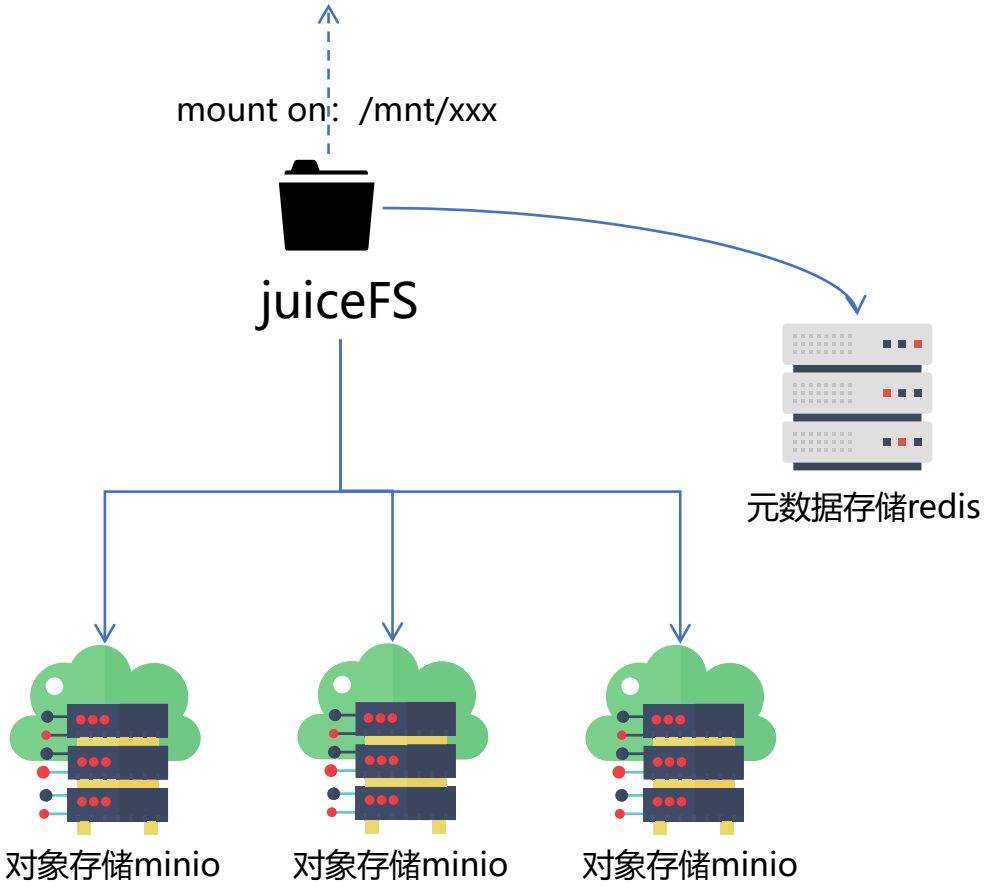
ClickHouse TTL机制

```
engine = MergeTree ORDER BY (event_type, repo_name, created_at)
    TTL
        created_at TO VOLUME 'v_hot',
    log_time + toIntervalDay(7) TO VOLUME 'v_cold',
    log_time + toIntervalDay(14)
```

- 7天内数据进入v_hot
- 14天内数据进入v_cold
- 14天以上数据删除

注意:该机制是作用于单机的。通过和raid控制器的配合可以大大提高单机数据库处理能力。

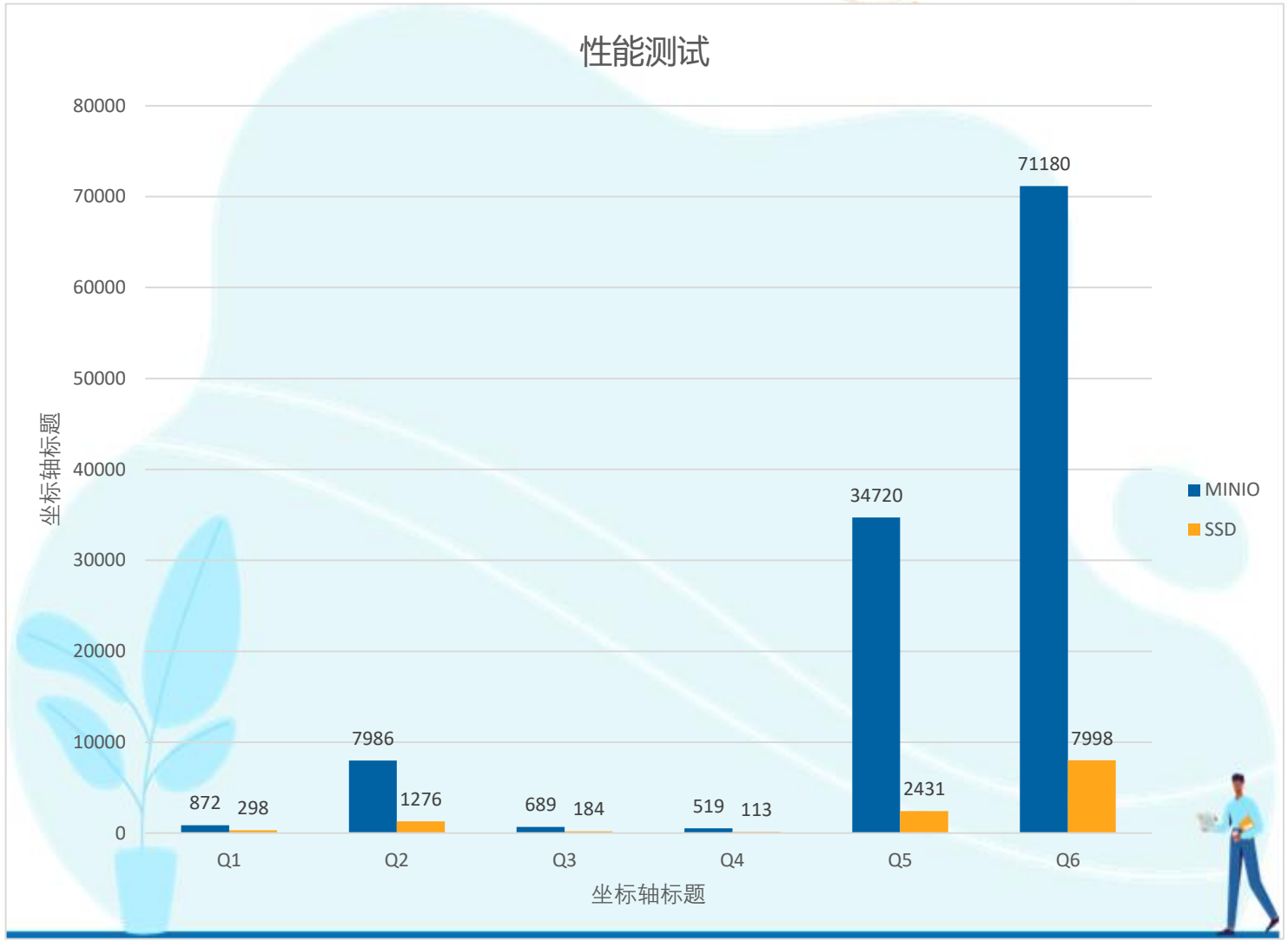
juiceFS + redis + minio



- 通过juiceFS将minio挂载为本地磁盘
- juiceFS的元数据存储在redis中

性能测试

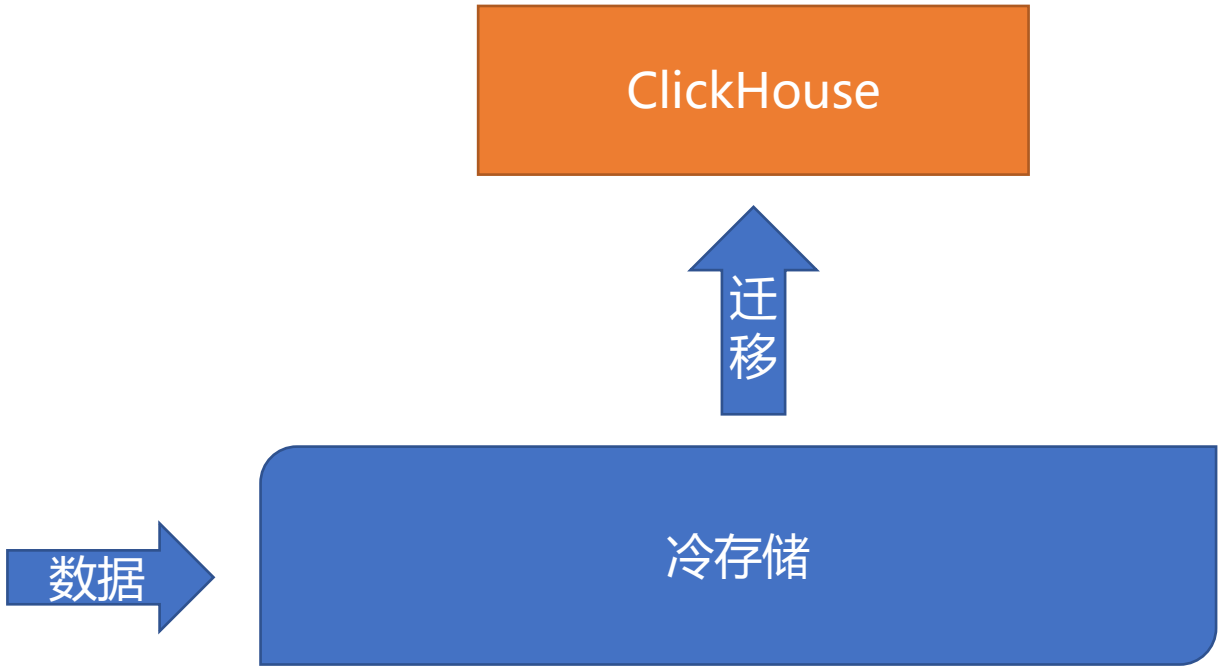
测试环境：
客户一天的数据：约400G，压缩后约80G
内存：128G
CPU：32vCPU
带宽：10Gbps
SSD + HDD（minio）



限制

- redis和juiceFS容易出单点故障
- 冷存储中的数据无法直接被其他引擎使用

冷—>热



- 数据先进入冷(相对)存储，例如hdfs/hive
- 通过数据迁移机制完成冷存储导入ClickHouse
- 很多已建设大数据系统的企业更倾向使用这种架构，可以将现有的投资重复利用，省去清洗等过程

ClickHouse的Integrations表引擎

```
engine = S3('  服务器地址  ',
    '用户名、密码', 'Native', 'xz');
```

支持引擎外部表：

1. HDFS
2. S3（minio兼容）
3. JDBC
4. ODBC
5. MySQL
6. MongoDB
7. Postgresql
8. sqlite
9. kafka
10. rabbitmq

方案一：直接查询外部表

优点

- 简单

限制

- 并发度不行（会导致多次传输数据）
- 受限于具体的外部表格式，可能需要遍历数据（无索引）

方案二：迁移数据

- 通过sql命令迁移：INSERT INTO new SELECT * FROM oldExternal
- 通过第三方工具迁移

SQL命令迁移注意事项—— too many parts

```
INSERT INTO s3.github_events_part_all SELECT *
FROM s3.github_events

Query id: 82c4fd80-77a7-4f1d-be5b-19dbc9252aef

↖ Progress: 565.25 thousand rows, 263.56 MB (187.94 thousand rows/s., 87.63 MB/s.)
0 rows in set. Elapsed: 3.050 sec. Processed 565.25 thousand rows, 263.56 MB (185.32 thousand rows/s., 86.41 MB/s.)

Received exception from server (version 21.8.4):
Code: 252. DB::Exception: Received from localhost:9000. DB::Exception: Too many partitions for single INSERT block (more than 100).
rtitions is a common misconception. It will lead to severe negative performance impact, including slow server startup, slow INSERT q
der 1000..10000. Please note, that partitioning is not intended to speed up SELECT queries (ORDER BY key is sufficient to make range
ile executing SinkToOutputStream.
```

当外部数据源返回的顺序与ClickHouse表配置的分区规则不一致时发生。

ClickHouse需要不停地创建新的分区，从而导致出现该问题。

解决方案：sql语句后加上order by xxx（按照分区规则）

SQL命令迁移注意事项—— Memory Limit

```
INSERT INTO s3.github_events_part_all SELECT *
FROM datasets.github_events
ORDER BY toYear(created_at) ASC

Query id: 45acc1c0-1ace-4008-928c-7ba642eecb22

→ Progress: 16.71 million rows, 7.86 GB (1.09 million rows/s., 512.52 MB/s.)
0 rows in set. Elapsed: 15.388 sec. Processed 16.71 million rows, 7.86 GB (1.09 million rows/s., 510.62 MB/s.)

Received exception from server (version 21.8.4):
Code: 241. DB::Exception: Received from localhost:9000. DB::Exception: Memory limit (for query) exceeded: would use 9.32 GiB
k): (while reading from part /mnt/data/clickhouse/data/datasets/github_events/all_687_2326_4/ from mark 24 with max_rows_to_r
```

这个是ClickHouse执行逻辑导致的，insert into x select中数据需要载入内存后才会被处理。

解决方案：
1. 调大ClickHouse的内存
2. sql语句后加上where，分批导入数据
3. 加上一些配置项：max_block……

解决思路

- 系统化、体系化的迁移方案
- 按照分区为单位分批迁移

DBMS4CK —— 解决上述问题的工具

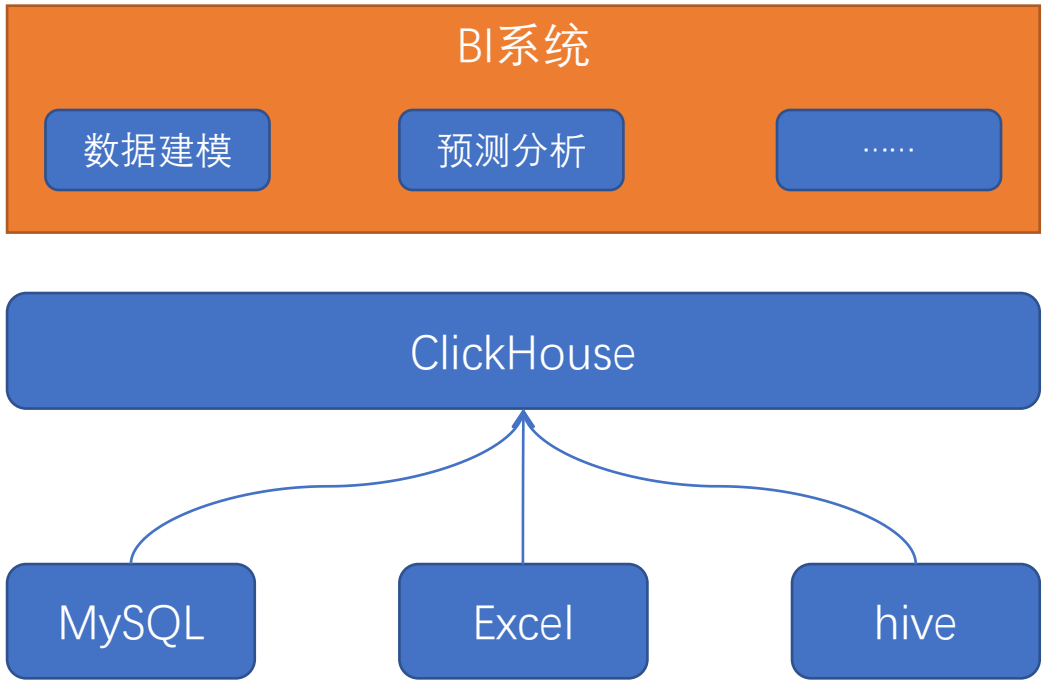
- ClickHouse的管理工具
- 具备集群安装、运维能力。一键安装集群
- 类似hue的sql编辑器
- notebook功能
- 执行sql的统计、分析、优化建议（建议增加跳数索引、增加投影等）
- 傻瓜式的ClickHouse独有功能的操作向导（外部表、联邦查询、insert into select计划）

HeatedMergeTree — 自动化表引擎

- 自研引擎，类似Distributed引擎
- 解决冷->热运维难的问题
- 解决proxy方案中无法应对复杂查询的情况
- 通过构建查询计划时分析AST，通过改变查询计划来实现
- 构建heatmap，自动将冷数据迁移进ck（以分区为单位）

案例

案例一：某客户BI架构

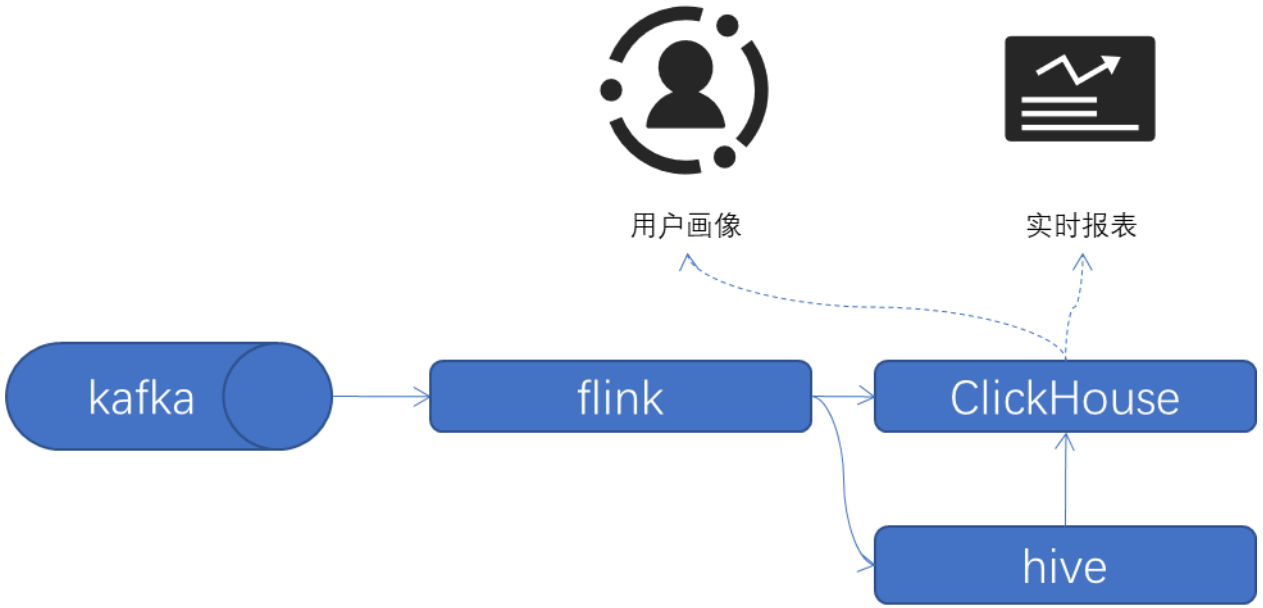


- 所有数据导入到ClickHouse中进行分析
- 上层的BI系统只需对接ClickHouse
- BI的交互式分析查询需求恰巧时ClickHouse的强项
- ClickHouse中的数据只用于分析查询，可以数据还在原始数据库中，可以随时删除重建。避开了ClickHouse更新数据的劣势

案例二：某iot客户时序数据分析

- 类kappa架构，使用ClickHouse取代了flink,降低了kafka中存储的数据量
- ClickHouse中约为百TB的数据量级
- iot每隔一段时间上报一次事件，**数据写入后不会修改**，只有查询分析的需求，符合ClickHouse的场景

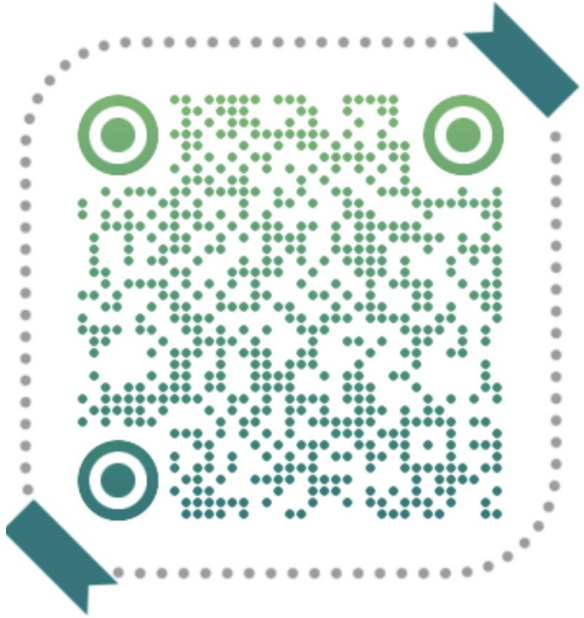
案例三：某零售客户实时分析



- 降低flink集群压力
- 指标使用sql进行描述，业务难度、运维难度低（使用flink的话需要重新部署任务）且flinksql支持的分析函数少
- 摊销ClickHouse的处理时间

总结&使用建议

- 避免点查、全表扫描，该加的二级索引要加上
- 宽表，越宽越好
- 分区粒度不能太小
- 善用物化视图、投影，慎用join
- 分布式表查询，本地表写入，用好代理
- 避免小批次量写入，否则会出现too many parts



《clickhouse架构与源码解析》


THANKS
时序
开源
自研
分布式
文档
Spark
HANA
SQL
OLAP
Oracle
Hadoop
MPP
DB2