趣头条基于Flink+ClickHouse构建实时数据平台

王金海
趣头条数据平台负责人

FLINK FORWARD # ASIA
实时即未来 # Real-time Is The Future

FLINK FORWARD

自我介绍

About me

01/ 9年互联网历练，先后在唯品会、饿了么负责大数据开发架构工作

02/ 现为趣头条数据中心平台负责人。负责大数据基础设施建设（spark、presto、flink、clickhouse）、数据产品化输出（qe、horizon、kepler）、团队建设

目录

Content

业务场景与现状分析

Business scenario and current situation analysis

Flink-to-Hive小时级场景

Flink-to-Hive

Flink-to-ClickHouse秒级场景

Flink-to-ClickHouse

未来发展与思考

Future

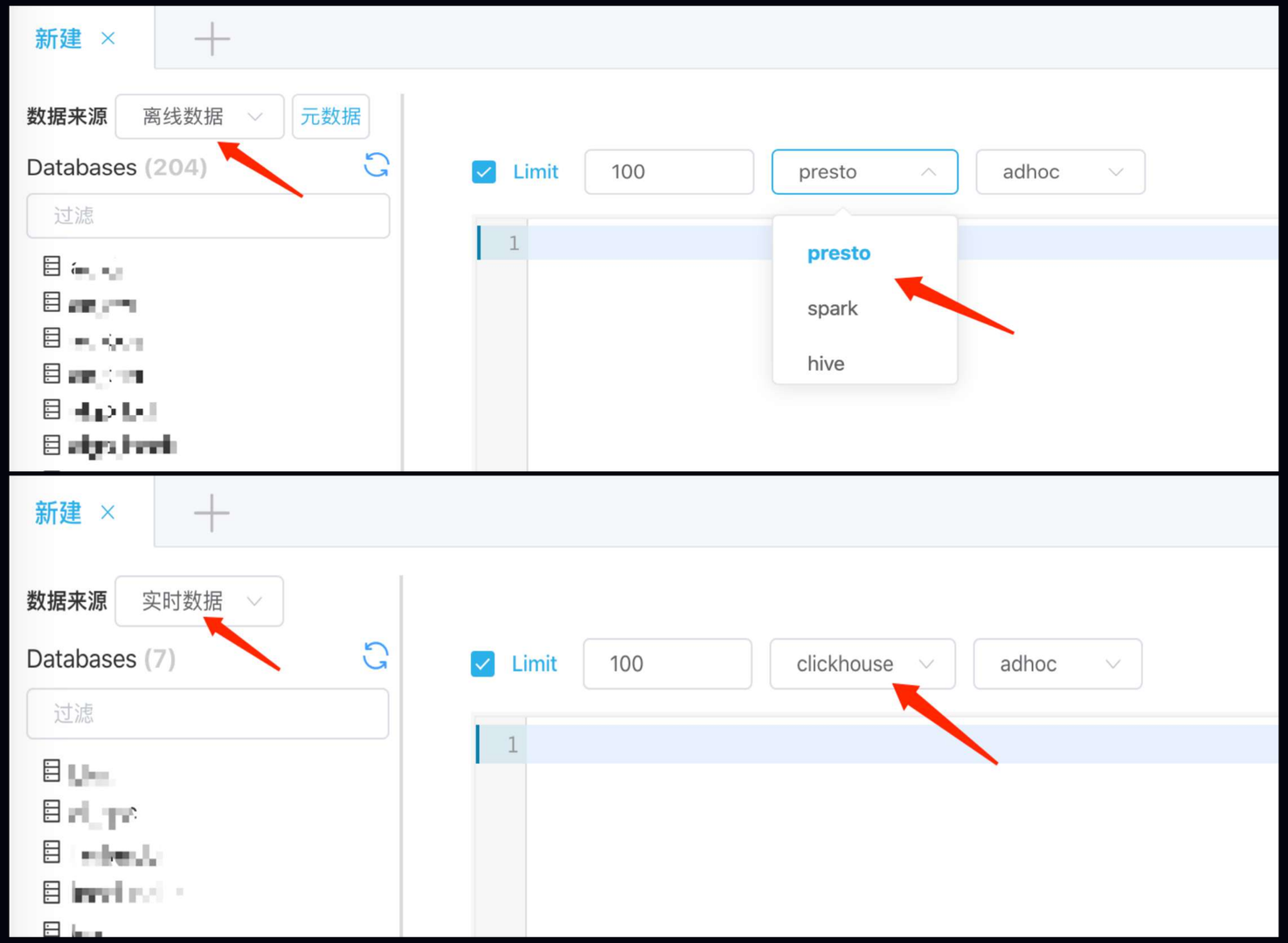
业务场景与现状分析

Business scenario and current situation Analysis

01

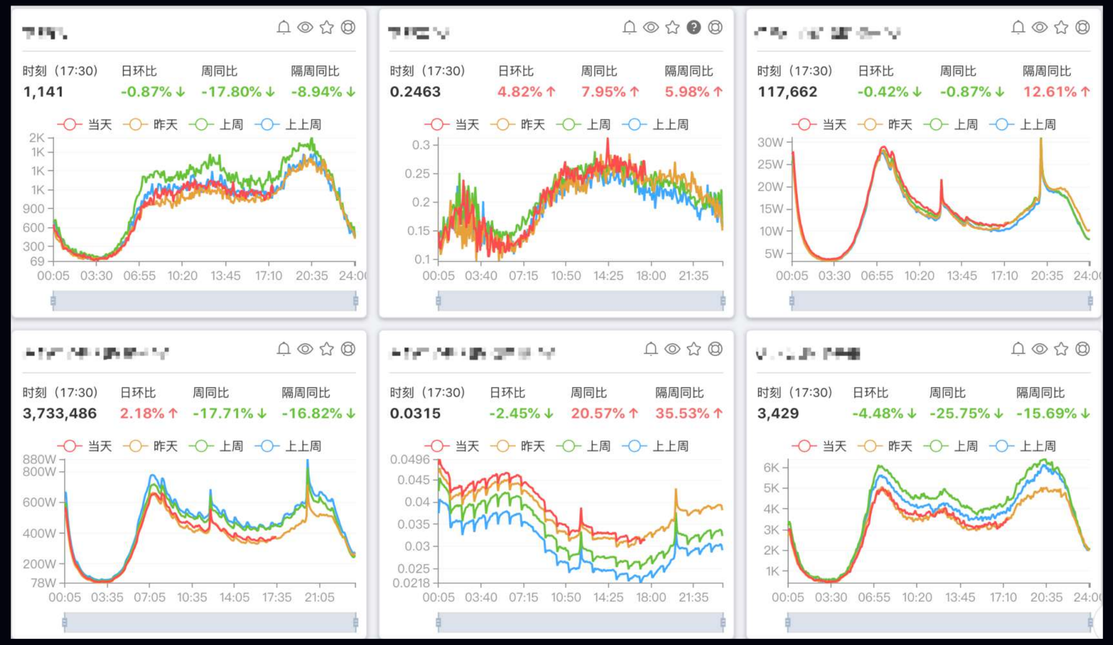
业务场景与现状分析

Business scenario and current situation Analysis



业务场景与现状分析

Business scenario and current situation Analysis



时段	当天（[illegible]）	昨天（[illegible]）	上周（[illegible]）	上上周（[illegible]）
17:35	226551	+1.09% 224104	-13.21% 261031	-12.60% 259206
17:30	229365	+2.34% 224112	-12.94% 263467	-10.18% 255371
17:25	232633	+5.69% 220102	-13.77% 269768	-8.90% 255370
17:20	229028	+1.66% 225280	-12.35% 261304	-10.28% 255283
17:15	221130	-4.29% 231042	-13.11% 254494	-14.46% 258511

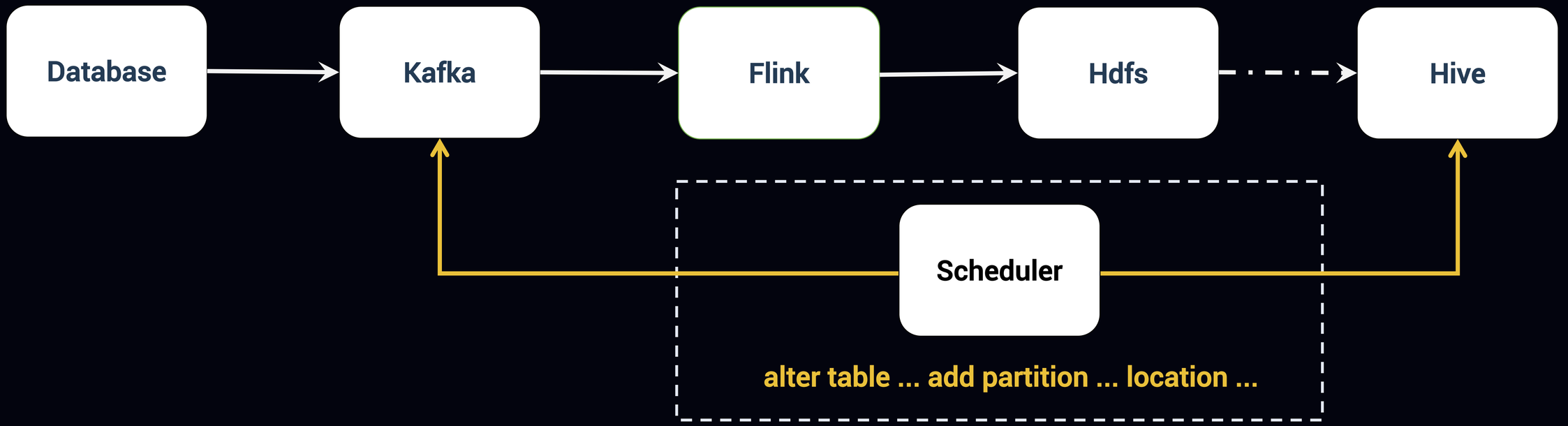
Flink-to-Hive小时级场景

Flink-to-Hive

02

小时级实现架构图

Hour level architecture



实现原理

StreamingFileSink API

```
StreamingFileSink.forBulkFormat(new Path("hdfs://data/xxx.db/yyyy/"),
        ParquetAvroWriters.forGenericRecord(schema))
        .withBucketAssigner(new EventTimeBucketAssigner())
        .withBucketCheckInterval(10 * 60 * 1000L)
        .build();
```

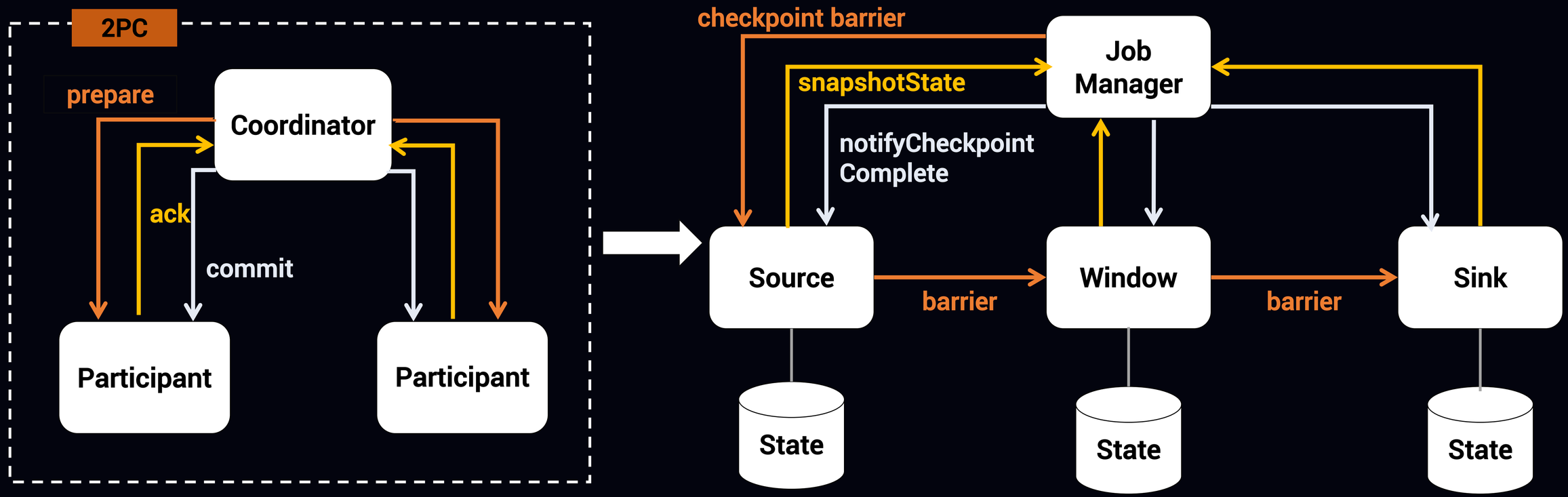
1. forBulkFormat支持avro、parquet格式

2. withBucketAssigner自定义按数据时间分桶

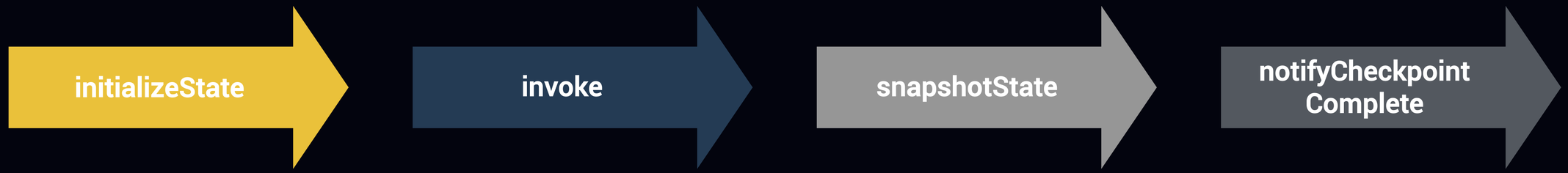
3. OnCheckpointRollingPolicy

4. Exactly-Once语义实现

Exactly-Once



StreamingFileSink implements CheckpointedFunction, CheckpointListener



- commitPendingFile
- restoreInProgressFile
- truncate (Hadoop 2.7+)

- write

- in-progress to pending state
- length
- ListState

- pending to finished state
- rename

跨集群多nameservices

Cross-cluster multi-nameservices

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>dfs.nameservices</name>
        <value>stream,data</value>
        <final>true</final>
    </property>

    <!-- stream实时集群的namenode ha配置  -->

    <!-- data离线集群的namenode ha配置 -->
</configuration>
```

多用户写入权限

Multi-user write permission

HadoopFsFactory.java

```java
@Override
public FileSystem create(URI fsUri, String user) throws IOException {
    try {
        return UserGroupInformation.createRemoteUser(user).doAs(new
PrivilegedExceptionAction<FileSystem>() {
            @Override
            public FileSystem run() throws Exception {
                return create(fsUri);
            }
        });
    } catch (InterruptedException e) {
        throw new IOException(e);
    }
}
```

```java
StreamingFileSink.forBulkFormat(new Path("hdfs://data/xxx.db/yyyy/"),
        ParquetAvroWriters.forGenericRecord(schema))
        .withBucketAssigner(new EventTimeBucketAssigner())
        .withBucketCheckInterval(10 * 60 * 1000L)
        .withBucketUser("stream")
        .build();
```

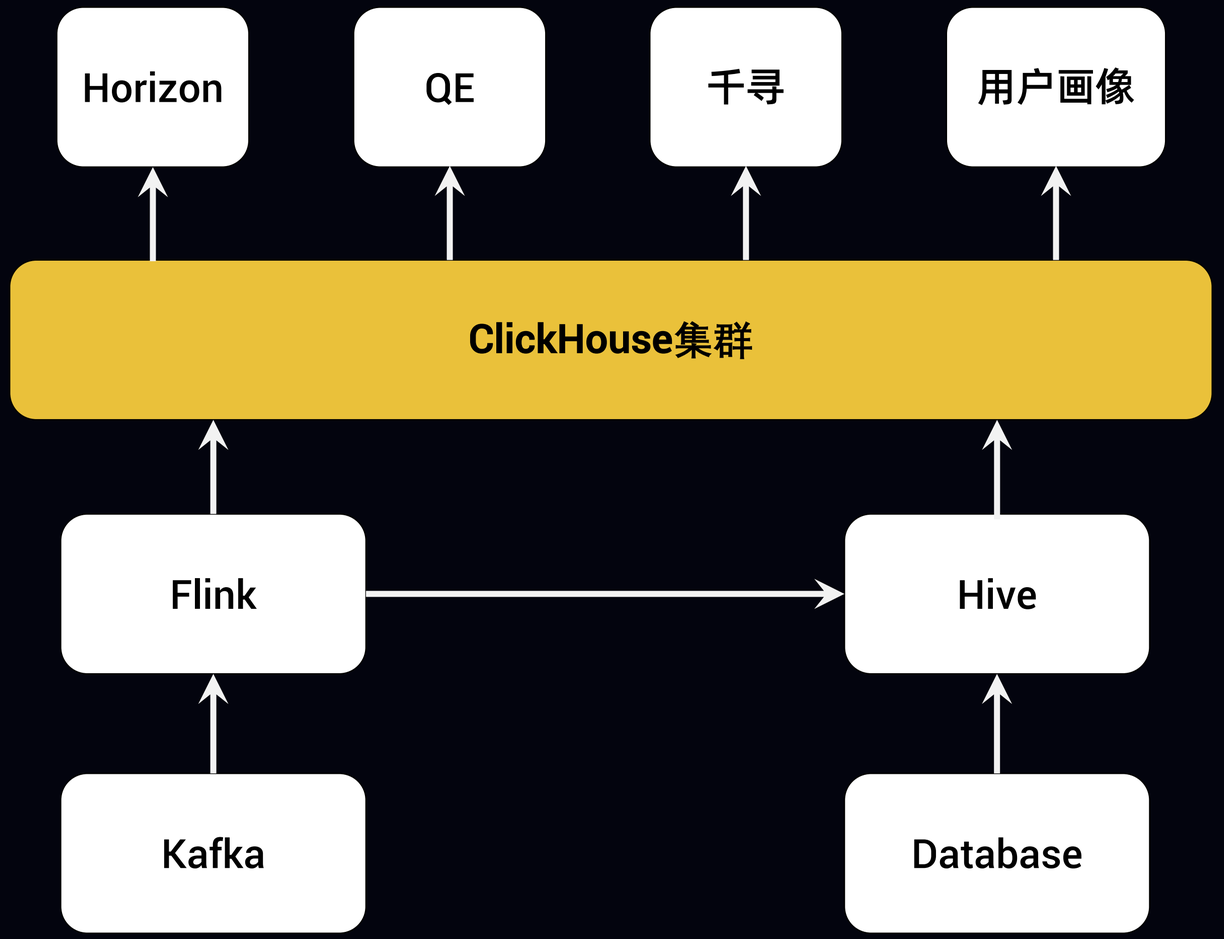
Flink-to-ClickHouse秒级场景

Flink-to-ClickHouse

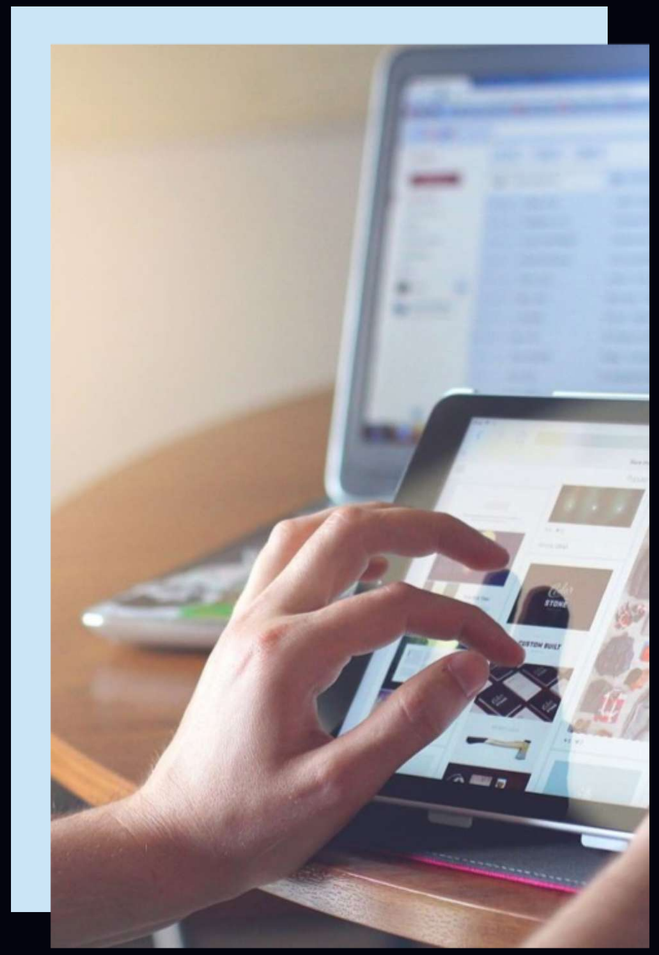
03

秒级实现架构图

Second level architecture



Why Flink+ClickHouse

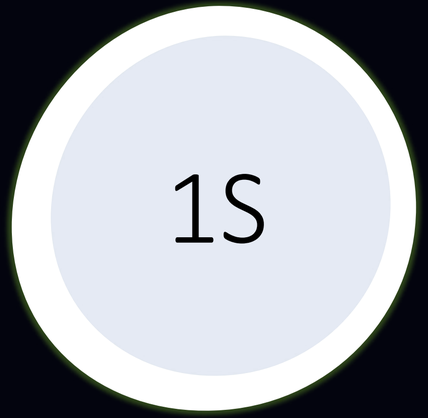


1. 指标实现支持sql化描述
2. 指标的上下线互不影响
3. 数据可回溯，方便异常排查
4. 计算快，一个周期内完成所有指标计算
5. 支持实时流，分布式部署、运维简单

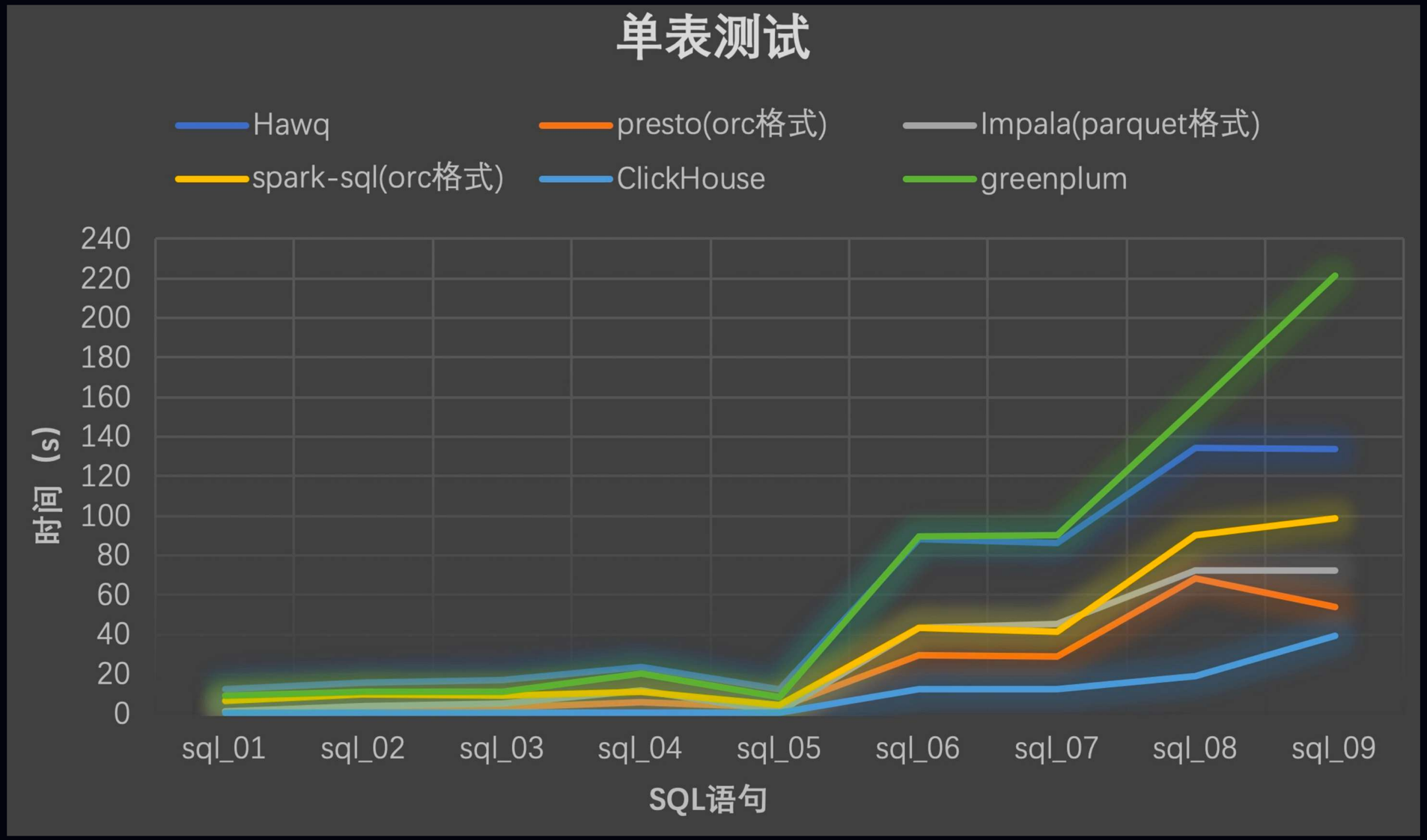
100+台32核128G
3.5T SSD

2000+亿/天

21+万次查询/天



80%查询1S内完成


单表测试
Hawq
presto(orc格式)
Impala(parquet格式)
spark-sql(orc格式)
ClickHouse
greenplum
240
220
200
180
160
140
120
100
80
60
40
20
0
时间（s）
sql_01
sql_02
sql_03
sql_04
sql_05
sql_06
sql_07
sql_08
sql_09
SQL语句

```
bjg-data-platform-clickhouse-01 :) select cmd, count(1) as cnt from qu[illegible]_local_20191106 gro
up by cmd order by cnt desc limit 10;

SELECT
    cmd,
    count(1) AS cnt
FROM qu[illegible]_20191106
GROUP BY cmd
ORDER BY cnt DESC
LIMIT 10
```

cmd	cnt
10[illegible]	5[illegible]480
30[illegible]	[illegible]208
50[illegible]	[illegible]750
15[illegible]	[illegible]91
10[illegible]	[illegible]6
40[illegible]	[illegible]2
59[illegible]	[illegible]5
80[illegible]	[illegible]3
20[illegible]	[illegible]6
80[illegible]	[illegible]

26亿
count + group by + order by
3.6S

```
10 rows in set. Elapsed: 3.624 sec. Processed 2.62 billion rows, 34.65 GB (722.05 million rows/s., 9.56 GB/s.)
```

Why ClickHouse so fast

01/ 列式存储 + LZ4、ZSTD数据压缩

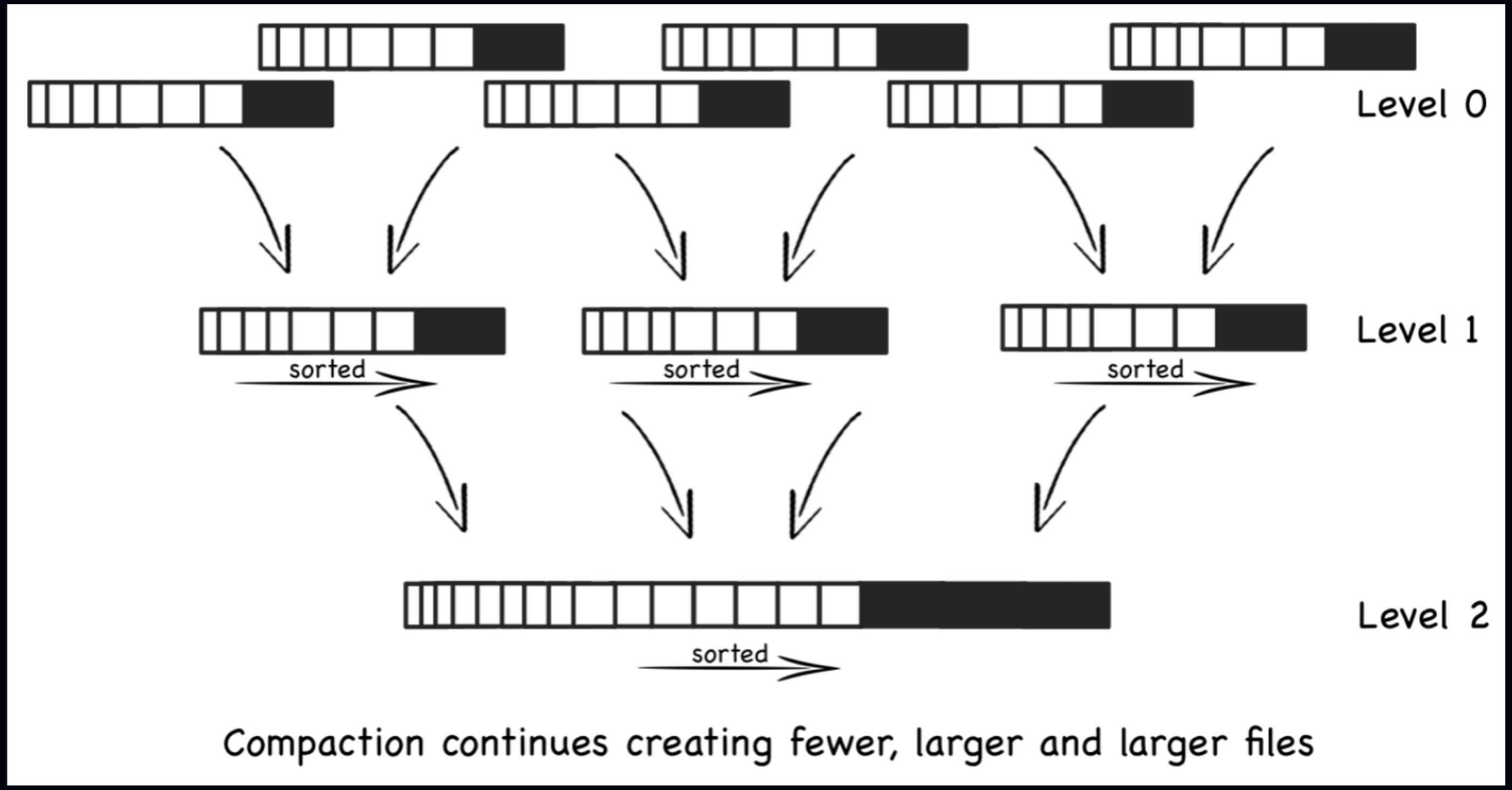
02/ 计算存储本地化 + 向量化执行

03/ LSM merge tree + Index

04/ SIMD + LLVM优化

05/ SQL语法、UDF完善

MergeTree


Level 0
Level 1
Level 2
sorted
sorted
sorted
sorted
Compaction continues creating fewer, larger and larger files

ClickHouse Connector

01

写Local table；
读Distributed table

02

5~10W batch；
5s interval

03

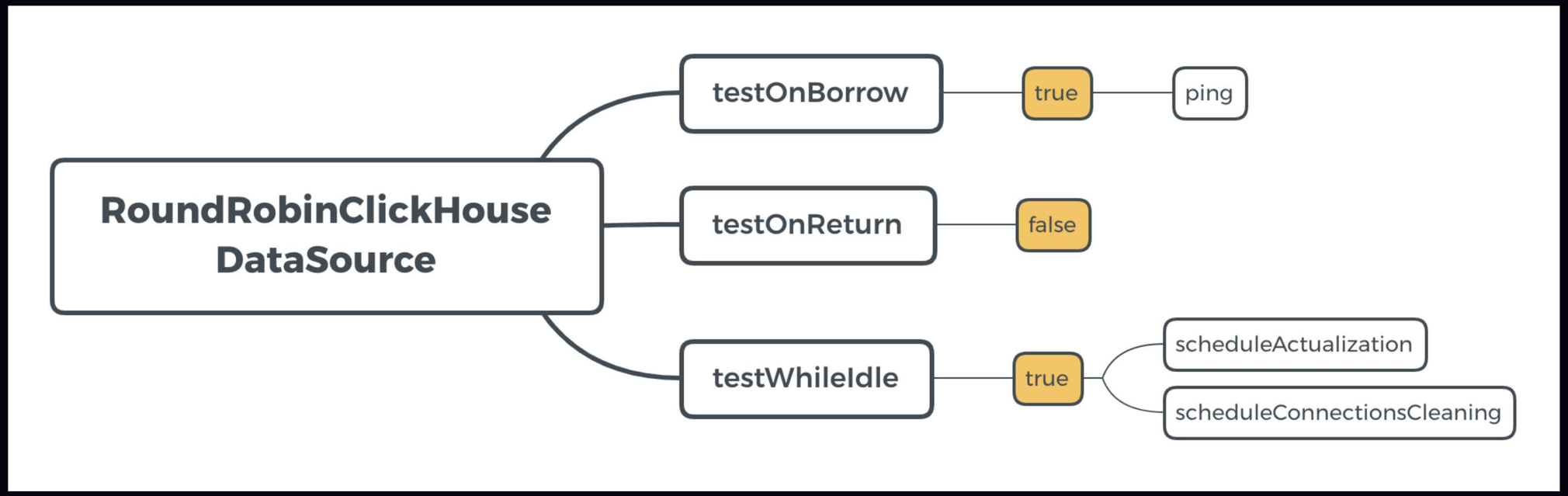
RoundRobinClickHouse
DataSource

BalancedClickHouseDataSource

01/ scheduleActualization

02/ scheduleConnectionsCleaning

RoundRobinClickHouseDataSource



Backfill

01/ Flink任务小时级容错

单击此处添加文本具体内容

02/ ClickHouse集群小时级容错

单击此处添加文本具体内容

未来发展与思考

New Future on Cloud

04

- Connectors SQL
- Delta lake


THANKS