Apache Flink

基于Iceberg的湖仓一体架构实践

邸星星/汽车之家实时计算平台负责人

Apache Flink x Iceberg Meetup · 上海站

CONTENT

目录 >>

#1 数据仓库架构升级的背景

严禁商业用途

基于Hive的数据仓库的痛点

不支持ACID

1、不支持Upsert场景

2、不支持Row-level delete，数据修正成本高

时效性难以提升

1、数据难以做到准实时可见

2、无法增量读取，无法实现存储层面的流批统一

3、无法支持分钟级延迟的数据分析场景

Table Evolution

1、写入型Schema，对schema变更支持不好

2、Partition Spec变更支持不友好

严禁商业用途

湖仓一体

引用贾扬清的介绍：

湖仓一体的意义就是说我不需要看见湖和仓，

数据有着打通的元数据的格式，它可以自由的流动，

也可以对接上层多样化的计算生态

严禁商业用途

Iceberg关键特性

Apache Iceberg is an open table format for huge analytic datasets

ACID

不会读到不完整的commit
基于乐观锁支持并发commit
Row-level delete，支持upsert

增量快照机制

Commit后数据即可见（分钟级）
可回溯历史快照

开放的表格式

数据格式：parquet、orc、avro
计算引擎：Spark、Flink、Hive、Trino/Presto

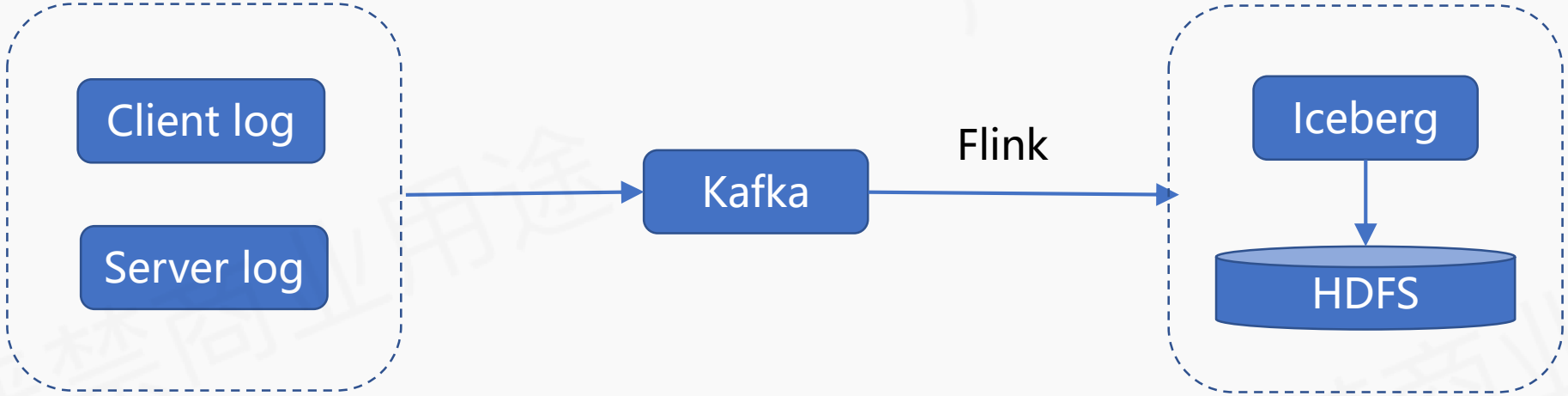
流批接口支持

支持流、批写入
支持流、批读取

#2 基于Iceberg的湖仓一体架构实践

严禁商业用途

Append流入湖的链路



严禁商业用途

Flink SQL入湖链路打通

Flink1.11 + Iceberg 0.11

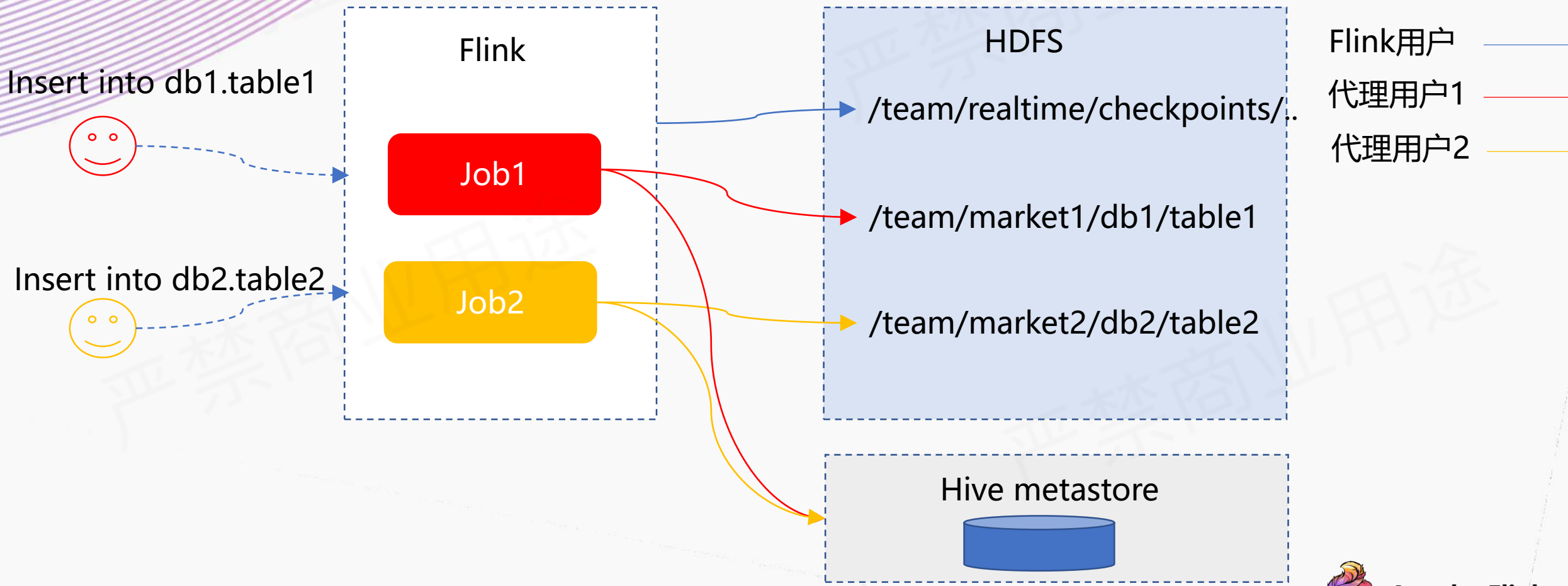
- 对接iceberg catalog：
  - Meta server 增加对iceberg catalog的支持
  - SQL SDK 增加iceberg catalog支持

- 平台开放iceberg表的管理功能

严禁商业用途

入湖-支持代理用户

对接现有预算体系、权限体系



严禁商业用途

入湖-支持代理用户

1、增加Table级别配置：'iceberg.user.proxy'='targetUser’

- 启用superuser
- 团队账号鉴权

用户申请团队权限

- 用户需要预先在离线平台申请hadoop团队账号

用户登录实时平台

- 用户登录AutoStream平台后，调用团队账号管理接口获取当前用户关联的团队账号

配置代理用户

- 在用户为任务设置代理用户时做鉴权：'iceberg.user.proxy' = '团队账号'

严禁商业用途

入湖-支持代理用户

2、访问HDFS时启用代理用户：

```
public static FileSystem getFs(Path path, Configuration conf) {
  try {
    String proxyUser = conf.get(ConfigProperties.ICEBERG_USER_PROXY);
    if (proxyUser != null) {
      UserGroupInformation ugi = UserGroupInformation.createProxyUser(proxyUser, UserGroupInformation.getLoginUser());
      return ugi.doAs((PrivilegedExceptionAction<FileSystem>) () -> path.getFileSystem(conf));
    } else {
      return path.getFileSystem(conf);
    }
  } catch (IOException e) {
    throw new RuntimeIOException(e, "Failed to get file system for path: %s", path);
  } catch (InterruptedException e) {
    throw new RuntimeException(e);
  }
}
```

Apache Flink

入湖-支持代理用户

3、访问Hive Metastore时指定代理用户

- 参考Spark的相关实现：
  org.apache.spark.deploy.security.HiveDelegationTokenProvider
- 动态代理HiveMetaStoreClient，使用代理用户访问 hive metastore

严禁商业用途

Flink SQL 入湖示例

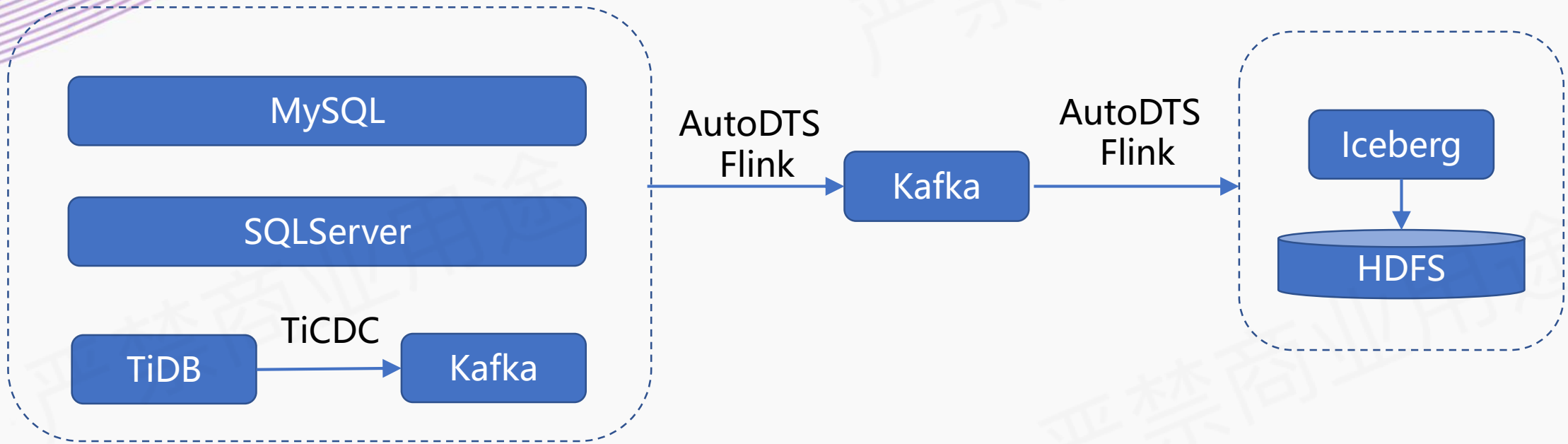
DDL + DML

```
CREATE TABLE ods.iceberg1 (
    `id`                string,
    `dt`                string,
    `xxx`               string
   PARTITIONED BY (`dt`)
) with('type'='ICEBERG',
'engine.hive.enabled'='true', // 支持hive查询
'read.split.target-size'='1073741824', // 减少split数,提升查询效率
'write.format.default'='parquet',
'iceberg.user.proxy'='someUser', // 代理用户设置
);

insert into ods.iceberg1
select `id`, `dt`, `xxx`
from kafka_ods.kafka_table1;
```

严禁商业用途

CDC数据入湖链路



严禁商业用途

Flink SQL CDC入湖链路打通

Flink1.11 + Iceberg 0.11

- 改进Iceberg sink：
  - Flink 1.11版本为AppendStreamTableSink，无法处理cdc流，修改并适配

- 表管理
  - 支持Primary key（PR1978）
  - 开启V2版本： 'iceberg.format.version'='2'

严禁商业用途

CDC数据入湖-支持bucket

Upsert场景下，需要确保同一条数据写入到同一bucket下

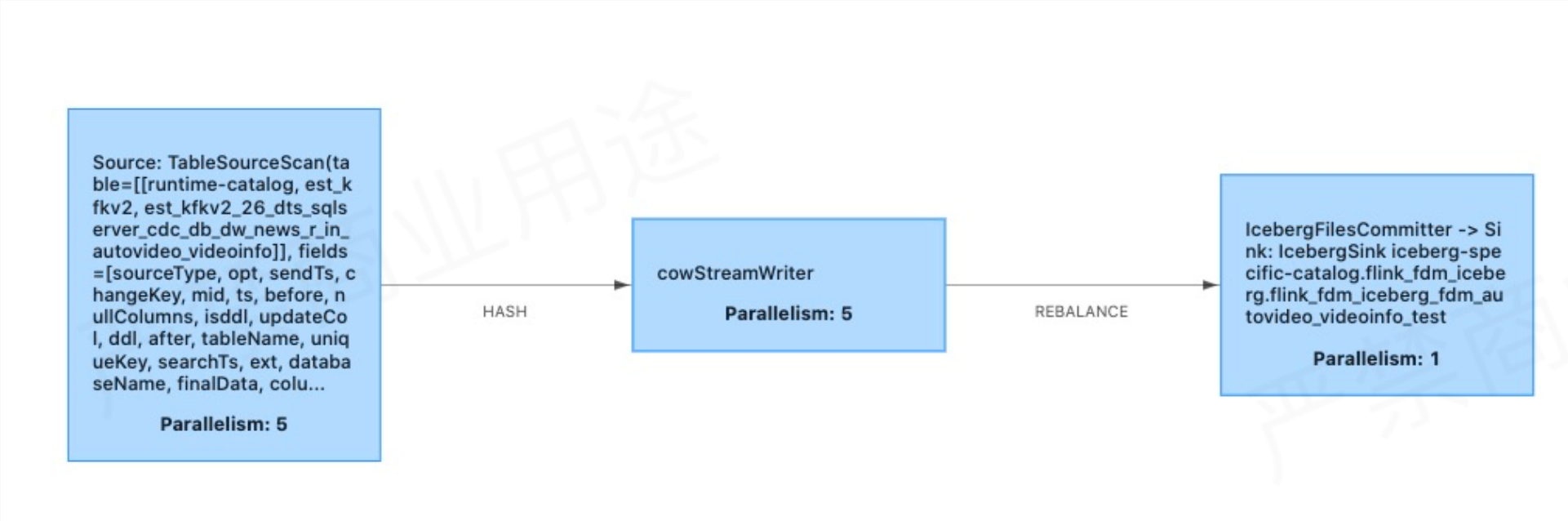
- 目前Flink SQL语法不支持声明bucket分区，通过配置的方式声明bucket：

```
'partition.bucket.source'='id',  // 指定bucket字段
'partition.bucket.num'='10',     // 指定bucket数量
```

严禁商业用途

CDC数据入湖：Copy-on-write sink

- StreamWriter多并行度写入
- FileCommitter单并行度顺序提交



严禁商业用途

CDC数据入湖：Copy-on-write sink

合理设置bucket数、无需引入小文件合并

- StreamWriter在snapshotState阶段多并行度写入
  - 增加Buffer
  - 写入前需要判断上次checkpoint已经commit成功
  - 按bucket分组、合并，逐个bucket写入

- FileCommitter单并行度顺序提交
  - table.newOverwrite()
  - flink.last.committed.checkpoint.id

严禁商业用途

CDC数据入湖：Copy-on-write sink

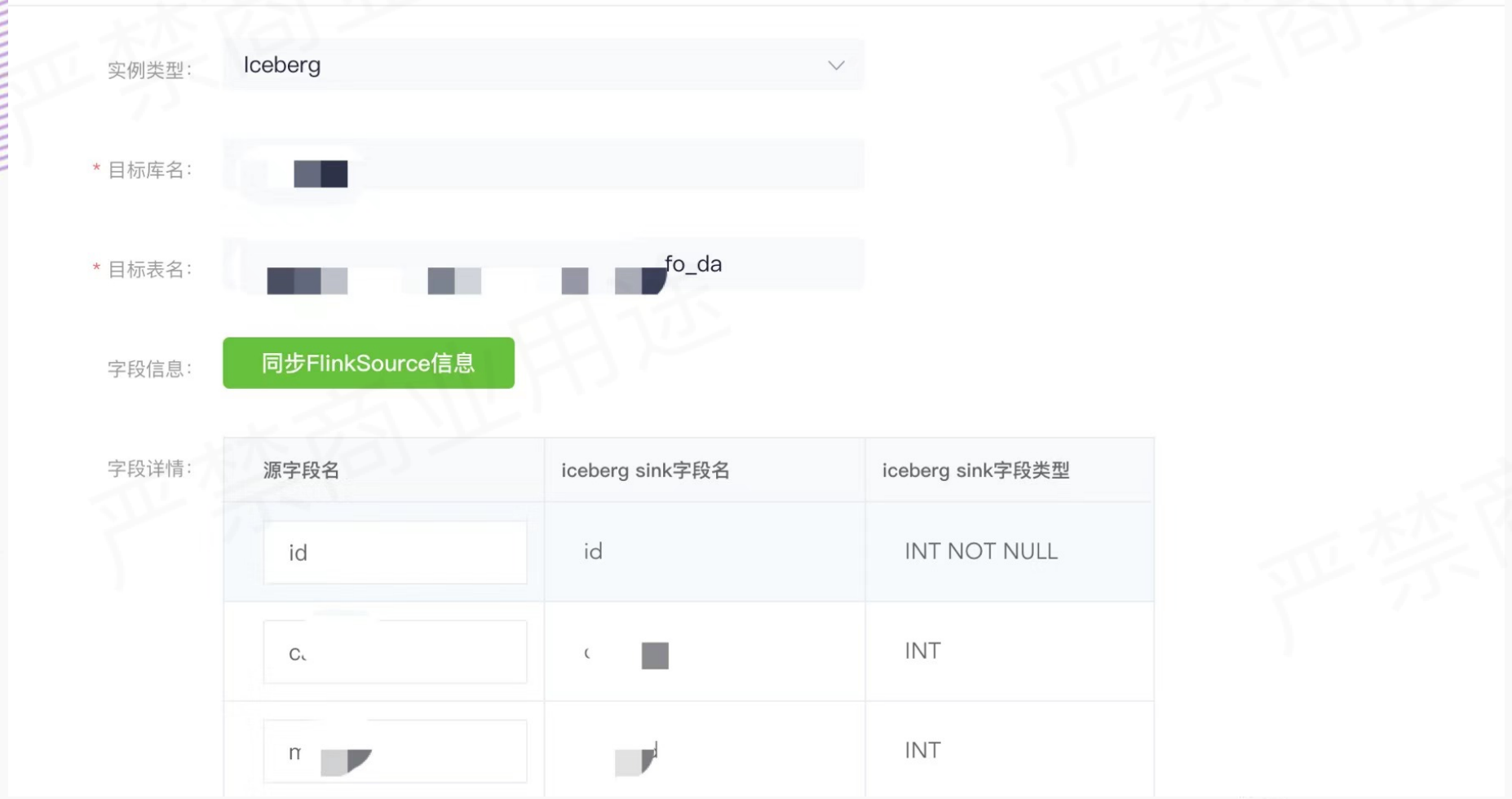
建表DDL

```
CREATE TABLE flink_iceberg_test (
    `id`                    string,
    ...
    PRIMARY KEY (id) NOT ENFORCED
) with('type'='ICEBERG',
'iceberg.format.version'='2',     // 声明需要开启v2格式
'engine.hive.enabled'='true',     // 支持hive
'read.split.target-size'='1073741824', // 减少split数,提升hive查询效率
'write.format.default'='parquet',
'iceberg.user.proxy'='someUser', // 代理用户设置
'partition.bucket.source'='id',  // 指定bucket字段
'partition.bucket.num'='10',      // 指定bucket数量
'write.distribution-mode'='hash',// 声明数据需要预先keyby
'write.sink.mode'='copy-on-write'// 声明copy-on-write 方式
)
```

Apache Flink

示例-CDC数据配置入湖

配置分发任务：



严禁商业用途

Apache Flink

Copy-on-write sink监控图表

入湖其他实践

减少empty commit

问题描述：
在上游kafka长期没有数据的情况下，每次checkpoint依旧会生成新的snapshot，导致大量的空文件和不必要的snapshot

解决方案（PR-2042）：
增加配置flink.max-continuous-empty-commits，在连续指定次数checkpoint都没有数据后才真正触发commit，生成snapshot

记录watermark

问题描述：
目前Iceberg表本身无法直接反映数据写入的进度，离线调度难以精准触发下游任务

解决方案（PR-2109）：
在commit阶段将flink的watermark记录到iceberg表的properties中，可直观的反映端到端的延迟情况，同时可以用来判断分区数据完整性，用于调度触发下游任务

删表优化

问题描述：
删除iceberg可能会很慢，导致平台接口相应超时。
因为iceberg是面向对象存储来抽象IO层的，没有快速清除目录的方法

解决方案：
扩展FileIO，增加deleteDir方法，在HDFS上快速删除表数据

严禁商业用途

小文件合并及数据清理

定期为每个表执行批处理任务（spark3）

#1

定期合并新增分区的小文件：
rewriteDataFilesAction.execute();
仅合并小文件，不会删除旧文件

#2

删除过期的snapshot，清理元数据及数据文件：
table.expireSnapshots().expireOlderThan(timestamp).commit();

#3

清理orphan文件，默认清理3天前，且无法触及的文件：
removeOrphanFilesAction.olderThan(timestamp).execute();

严禁商业用途

计算引擎-Flink

实时平台的核心计算引擎，目前主要支持数据入湖场景

#1

数据准实时入湖：

Flink和Iceberg在数据入湖方面集成度最高，Flink社区主动拥抱数据湖技术

#2

平台集成：

AutoStream 引入 IcebergCatalog，支持通过SQL建表、入湖

AutoDTS 支持将MySQL、SQLServer、TiDB表配置入湖

#3

流批一体：

在流批一体的理念下，Flink的优势会逐渐体现出来

严禁商业用途

Apache Flink

计算引擎-Hive

支持Hive 2.0.1

问题描述：
Iceberg默认支持Hive 2.3.7版本，不兼容我们的2.0.1版本

解决方案：
1、升级hive客户端的hive-exec 包，将SearchArgumentImpl 及相关类改为public class

2、修改Hive依赖版本，解决编译错误

默认忽略大小写

问题描述：
在0.10版本中，如果Iceberg表中包含大写字段，Hive 查询时会报错

解决方案（ PR-2053 ）：
在InputFormatConfig中增加配置：
iceberg.mr.case.sensitive
mr 默认为true（大小写敏感），Hive查询默认忽略大小写

Map数过多

问题描述：
将原hive表改为iceberg表后，使用hive进行查询，map数比直接查询hive表大很多

解决方案：
1、修改iceberg表的配置：
'read.split.target-size'='536870912'

2、升级spark合并小文件版本0.9 -> 0.11，0.9版本返回的orc文件大小不正确

计算引擎-Spark3

在SQL批处理层面Iceberg和Spark3集成度更高

#1

定期小文件合并及meta信息查询：

SELECT * FROM prod.db.table.history

还可查看snapshots, files, manifests

#2

离线数据写入：

Insert into

Insert overwrite

Merge into

#3

分析查询:

主要支持日常的准实时分析查询场景

严禁商业用途

计算引擎：Trino/Presto

AutoBI 已经和Presto集成，用于报表、分析型查询场景

- Trino
  - 直接将Iceberg作为报表数据源
  - 需要增加元数据缓存机制：https://github.com/trinodb/trino/issues/7551
- Presto
  - 社区集成中：https://github.com/prestodb/presto/pull/15836

严禁商业用途

踩过的坑

访问Hive Metastore异常

问题描述：
HiveConf的构造方法的误用，导致hive客户端中声明的配置被覆盖，导致访问Hive metastore时异常

解决方案（PR-2075）：
修复HiveConf的构造，显示调用addResource方法，确保配置不会被覆盖：hiveConf.addResource(conf);

Hive metastore锁未释放

问题描述：
CommitFailedException: Timed out after 181138 ms waiting for lock xxx.
原因是hiveMetastoreClient.lock方法，在未获得锁的情况啊下，也需要显示unlock，否则会导致上面异常

解决方案（PR-2263）：
优化HiveTableOperations#acquireLock方法，在获取锁失败的情况下显示调用unlock来释放锁

元数据文件丢失

问题描述：
Iceberg表无法访问，报NotFoundException Failed to open input stream for file : xxx.metadata.json

解决方案（PR-2328）：
当调用hive metastore 更新iceberg 表的metadata_location超时后，增加检查机制，确认元数据未保存成功后再删除元数据文件

#3 总结与收益

严禁商业用途

总结

湖仓一体、流批融合的探索

- 湖仓一体
  - Iceberg支持Hive Metastore
  - 总体使用上与Hive表类似：相同数据格式、相同的计算引擎
- 流批融合
  - 准实时场景下实现流批统一：同源、同计算、同存储

严禁商业用途

业务收益

数据时效性提升

入仓延迟从2小时以上降低到10分钟以内；算法核心任务SLA提前2小时完成

准实时的分析查询

结合Spark3和Trino，支持准实时的多维分析查询

特征工程提效

提供准实时的样本数据，提高模型训练时效性

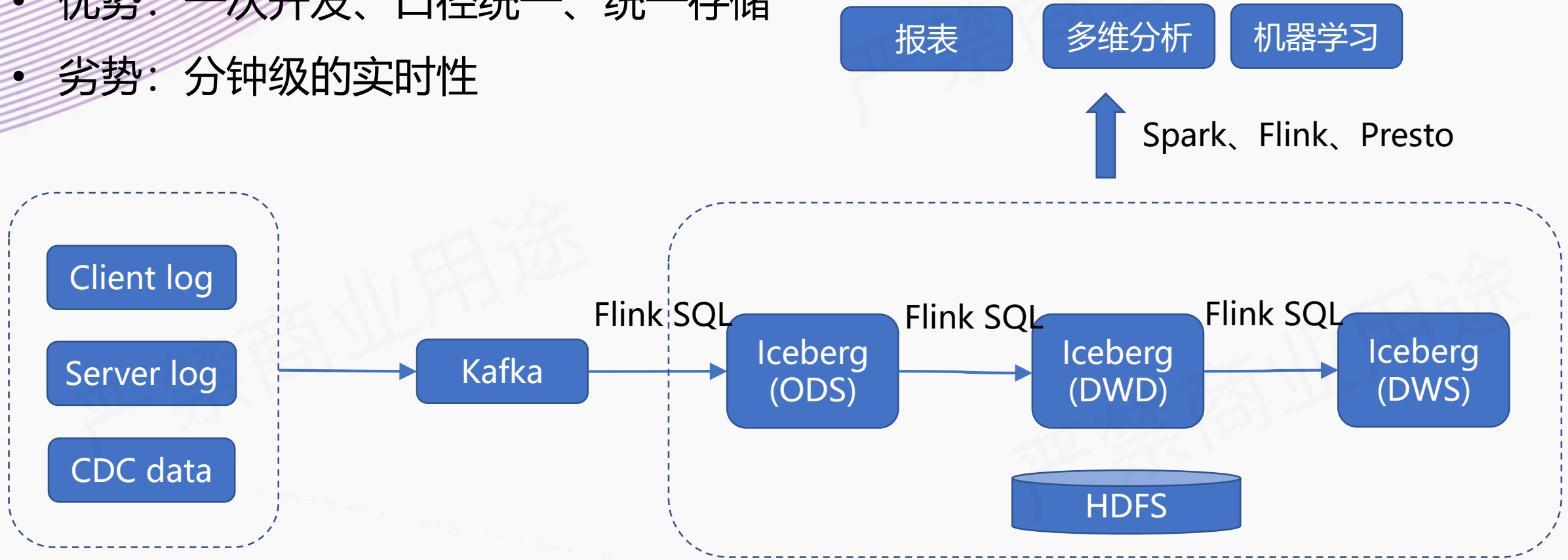
CDC数据准实时入仓

可以在数仓针对业务表做准实时分析查询

严禁商业用途

架构收益-准实时数仓

- 优势：一次开发、口径统一、统一存储
- 劣势：分钟级的实时性



#4 后续规划

严禁商业用途

后续规划

跟进 Iceberg 版本

全面开放V2格式，支持CDC数据的MOR入湖

建设准实时数仓

基于Flink通过Data pipeline模式对数仓各层表全面提速

流批一体

随着upsert功能的逐步完善，持续探索存储层面流批一体

多维分析

基于Presto/Spark3输出准实时多维分析

严禁商业用途

Apache Flink

Thanks

Apache Flink x Iceberg Meetup · 上海站