基于Flink+Hive构建流批一体准实时数仓

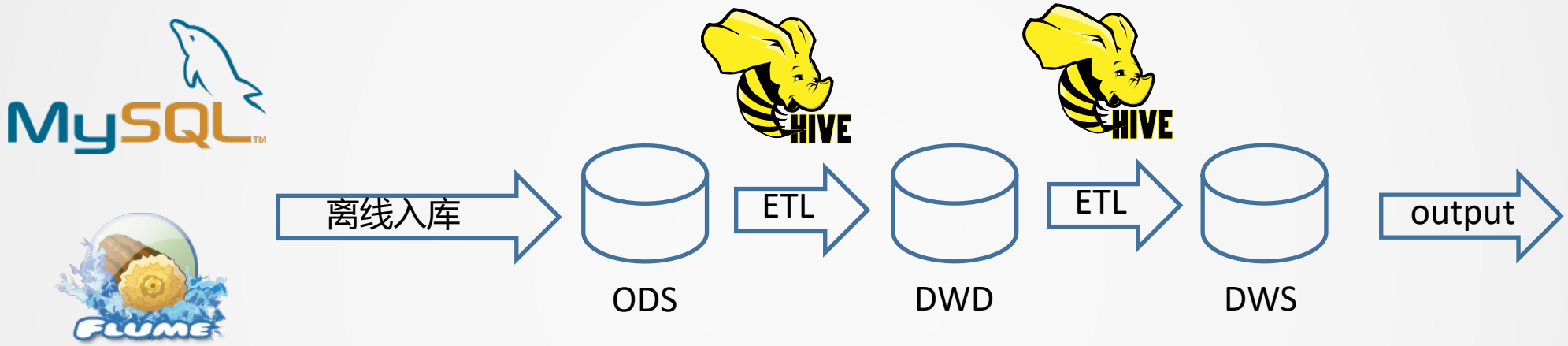
李劲松

分享大纲

- 离线数仓实时化的难点
- Flink 在流批一体上的探索
- 构建流批一体准实时数仓应用实践

离线数仓实时化的难点

离线数仓



- 离线数仓：T+1，H+1
- 持续计算：调度工具 + 血缘管理

第三方工具

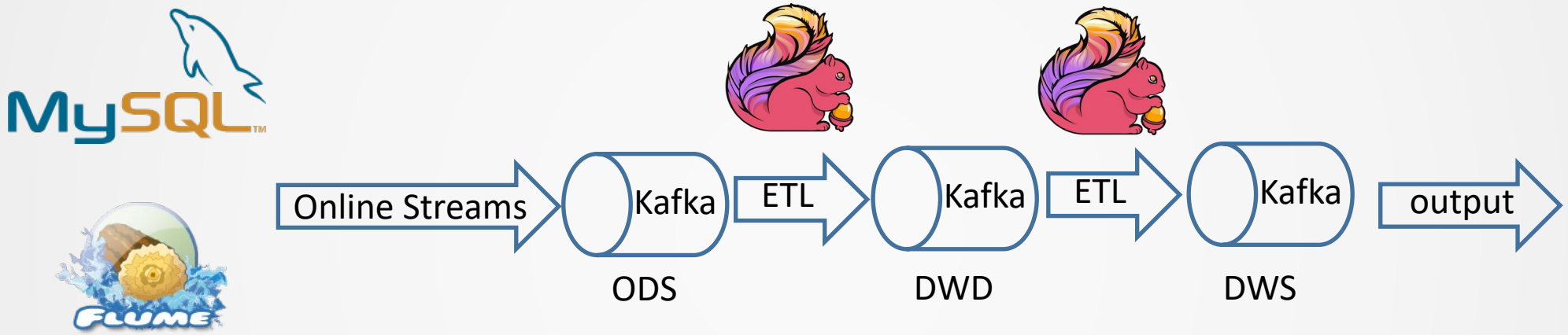
基于Flume的Hive入库

- 容错?
- 自定义逻辑?
- 扩展能力?

基于Oozie的Hive作业调度

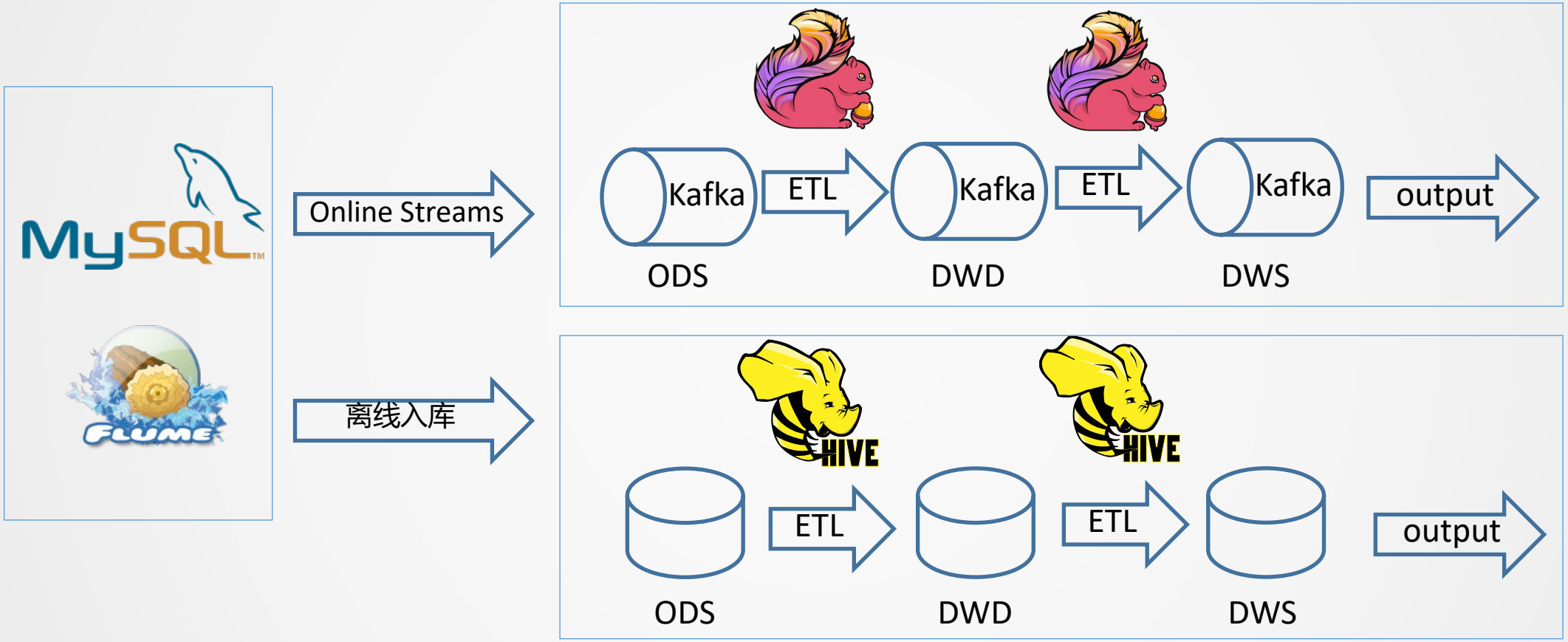
- 级联的计算延时
- 不准确的定时任务
- 复杂的血缘管理
- 优先级和负载管理

实时数仓



- 实时数仓：秒级延时
- 增量计算：流式增量计算
- 临时存储：历史数据丢失

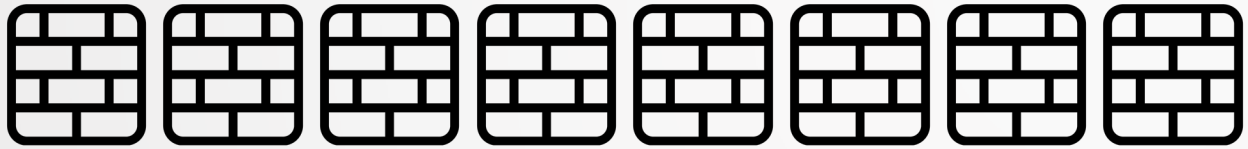
Lambda架构


MySQL
FLUME
Online Streams
离线入库
Kafka
ETL
Kafka
ETL
Kafka
output
ODS
DWD
DWS
HIVE
HIVE
ETL
ETL
output
ODS
DWD
DWS

Lambda架构

实时 Team

- 作业开发
- 计算资源
- 存储资源
- 集群维护



离线 Team

- 作业开发
- 计算资源
- 存储资源
- 集群维护

- 开发割裂感
  - 表结构不同
  - SQL语法不同
- 重复资源
  - 重复计算
  - 重复存储
- 集群维护
  - 组件不同
  - 计算引擎不同
- 数据一致性

实时需求

我们有这么多“真”实时需求吗?

“准”实时够了?

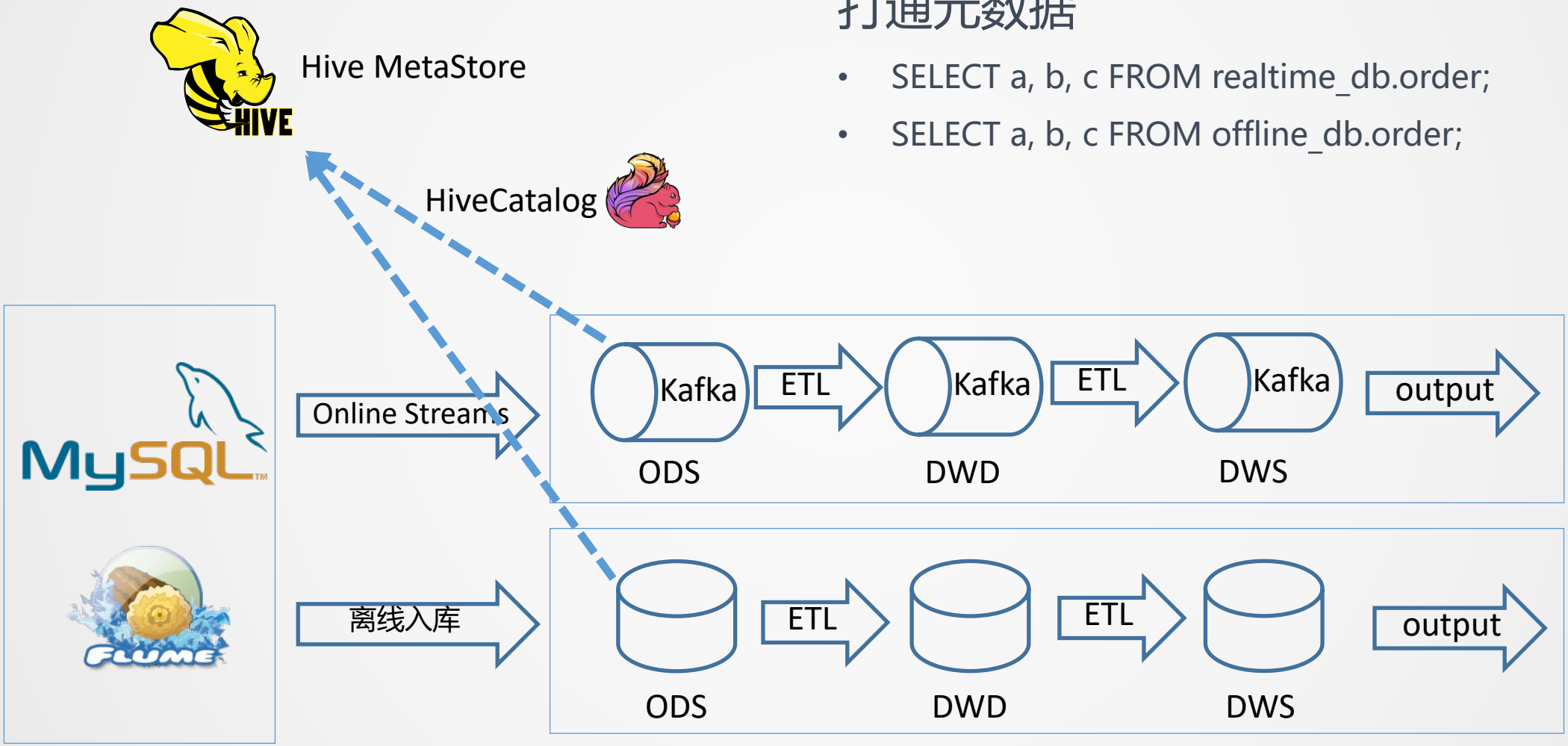
数据湖

- ACID：原子性
- 准实时批流一体
- UPDATE/DELETE

- 绑死Spark
- 迁移成本/学习成本
- 有多大好处?

Flink 在流批一体上的探索

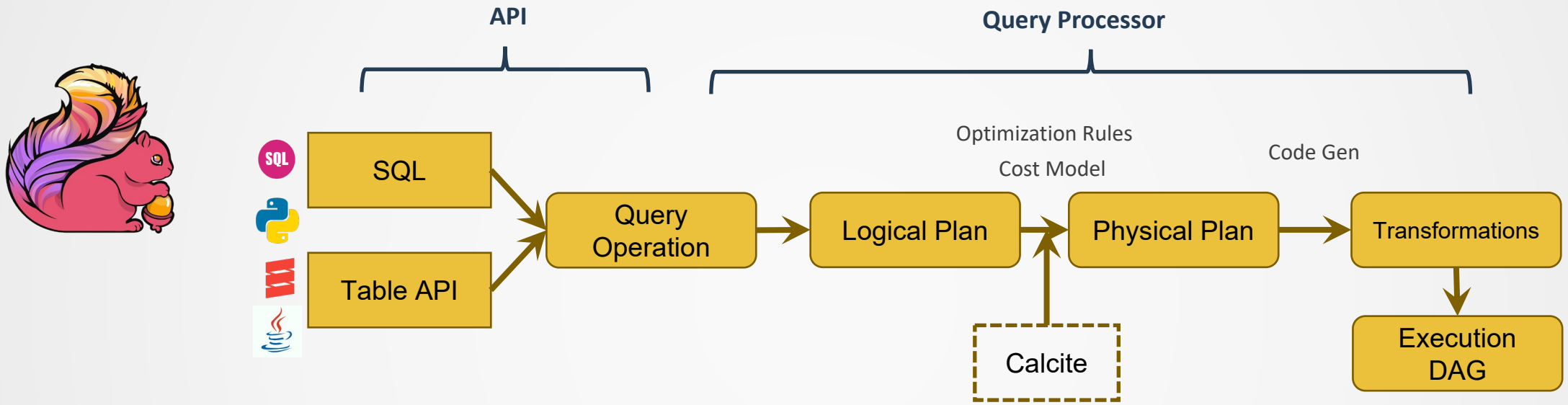
统一元数据



打通元数据

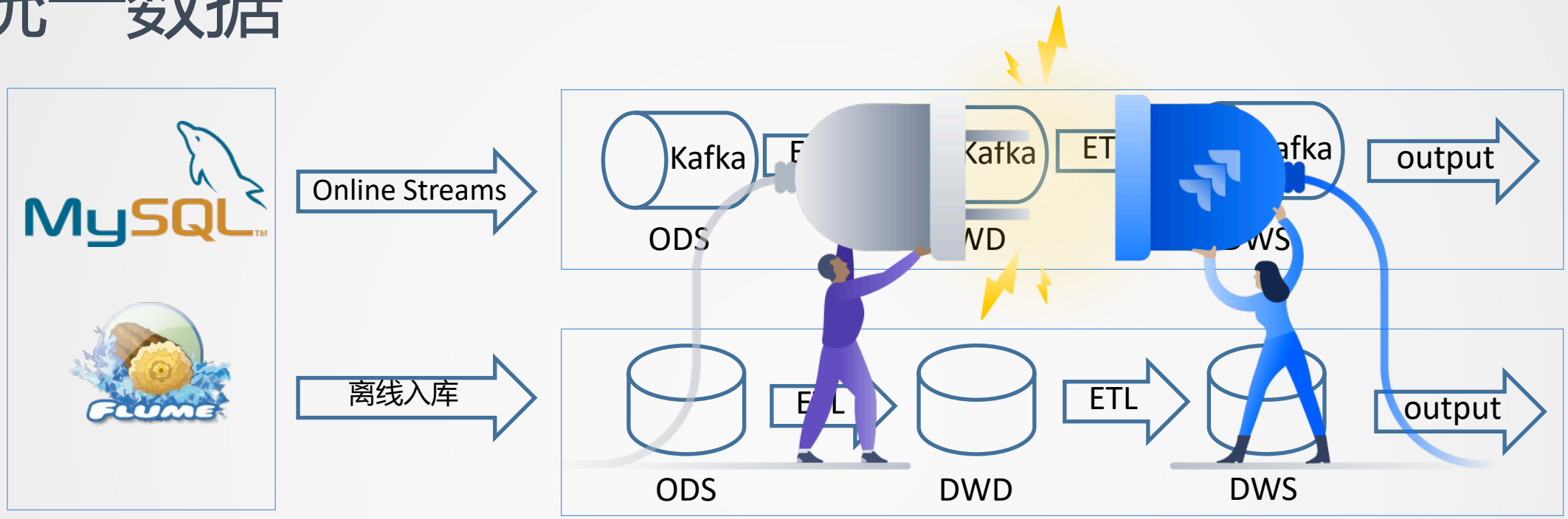
- SELECT a, b, c FROM realtime_db.order;
- SELECT a, b, c FROM offline_db.order;

统一计算引擎



- 一套SQL：Flink提供流批统一的ANSI-SQL语法
- 一个引擎：Flink流和批复用一套优化框架和Runtime执行

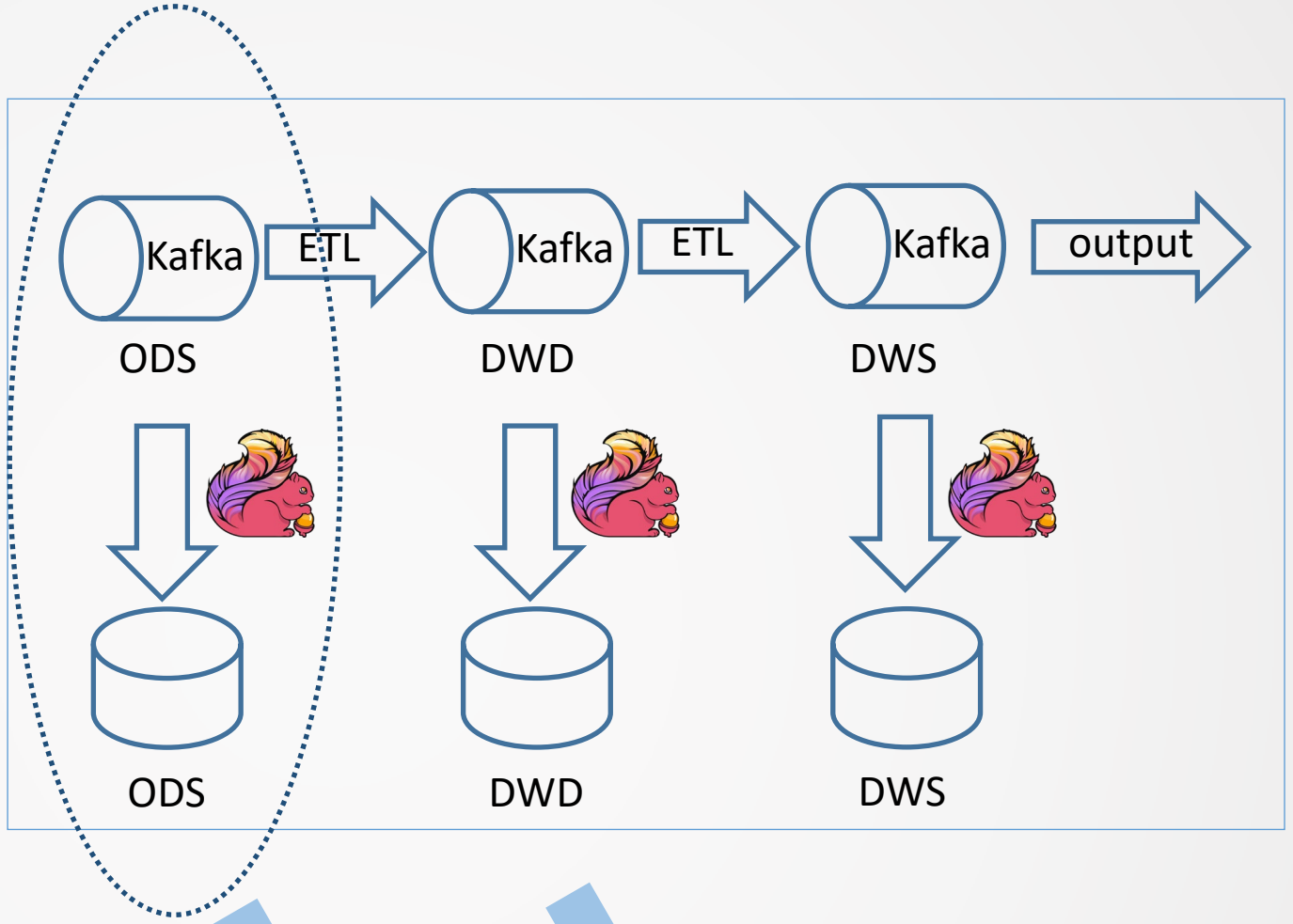
统一数据



- 统一数据，离线作为实时的历史
- 批数据准实时摄入，为Ad-hoc提供准实时输入
- 批计算由实时调度

准实时数据摄入

Online Streams



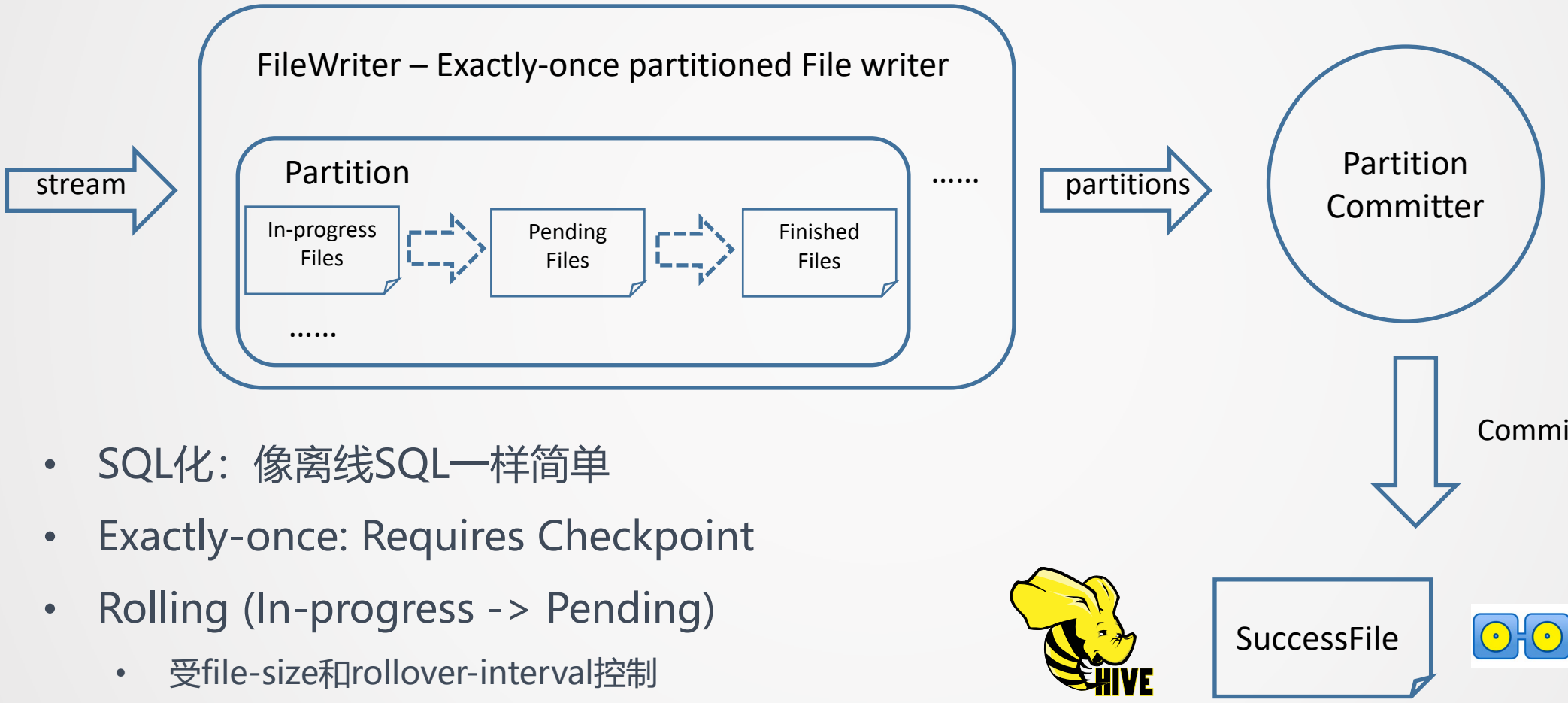
Flink Hive streaming sink

- 统一数据，减少不一致
- 减少重复计算
- 加快Hive数据准备，避免周期调度

A HybridTable?

实时能取代丰富的维表关联吗?

Hive streaming sink



- SQL化：像离线SQL一样简单
- Exactly-once: Requires Checkpoint
- Rolling (In-progress -> Pending)
  - 受file-size和rollover-interval控制
  - 受Checkpoint控制：频率太快会产生小文件

Partition Commit - WHEN

WHEN Partition Commit Trigger：什么时候提交分区

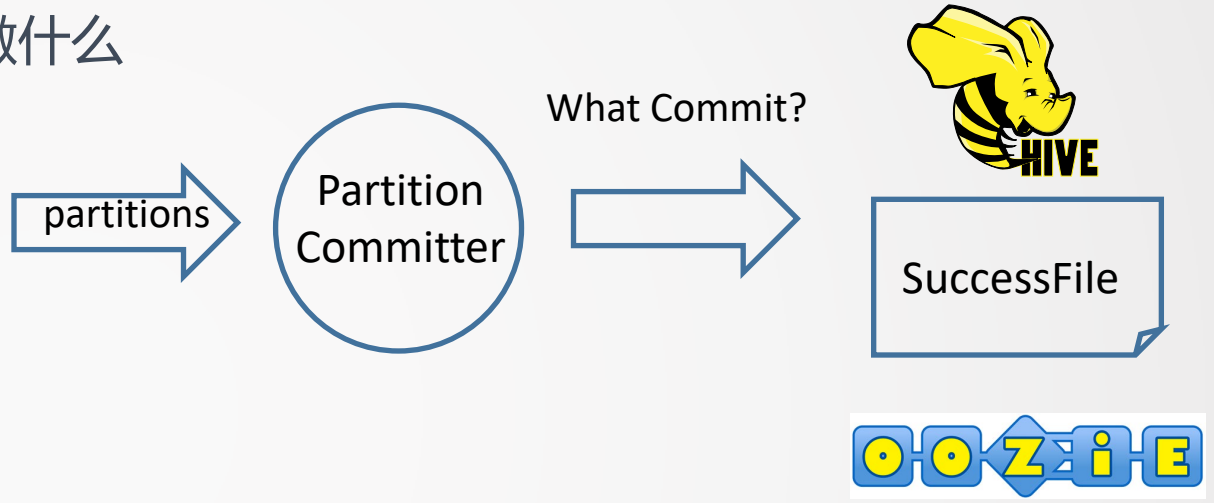
Process-time	Partition-time
根据处理时间决定提交的时间	根据Watermark+Partition字段时间决定提交的时间
优点：定义简单	优点：提交时间准确语义明确，数据完整，可以放心的交给下游消费了
缺点：提交分区时数据可能不完整	缺点：需求Input stream有EventTime

If Current time > partition start time + partition time size

- *Eg(小时级分区): Current time > 11点 + 1小时 (CommitDelay)*
- *Process-time: Current system time > partition出现的时间 + CommitDelay*
- *Partition-time: Current watermark > partition提取的时间 + CommitDelay*

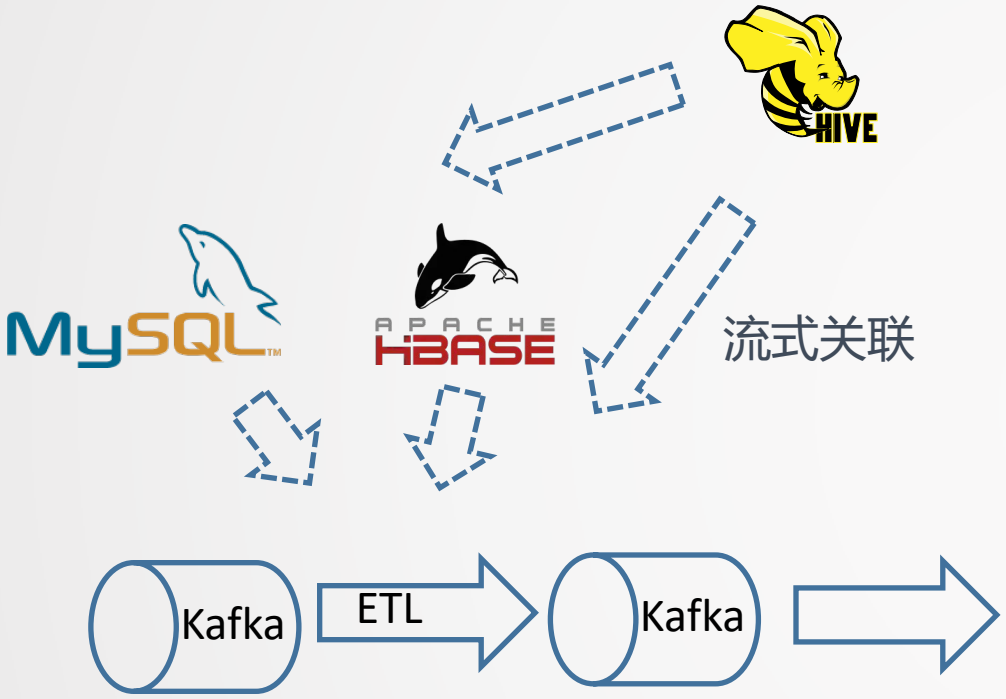
Partition Commit - WHAT

WHAT Partition Commit Policy：提交分区做什么

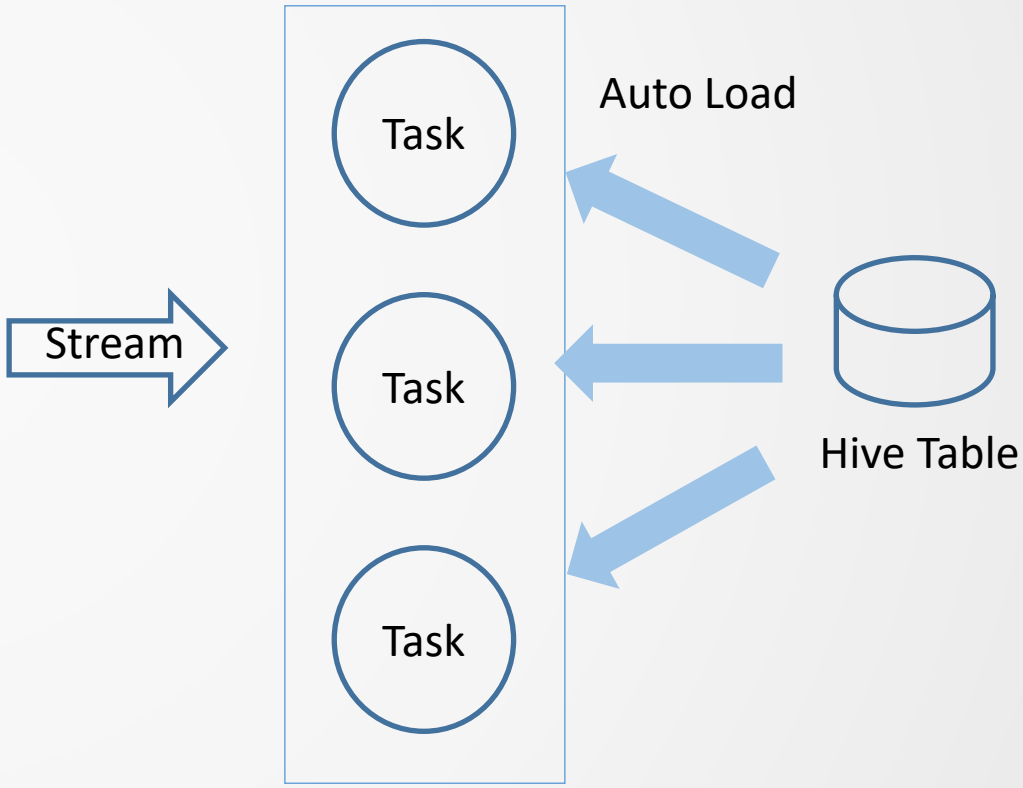


- 内置支持 Hive-MetaStore、SuccessFile
- Custom
  - 触发下游的调度
  - 触发 Hive 的 Statistic Analysis
  - 触发 Hive 的 小文件合并

实时维表关联 Hive tables

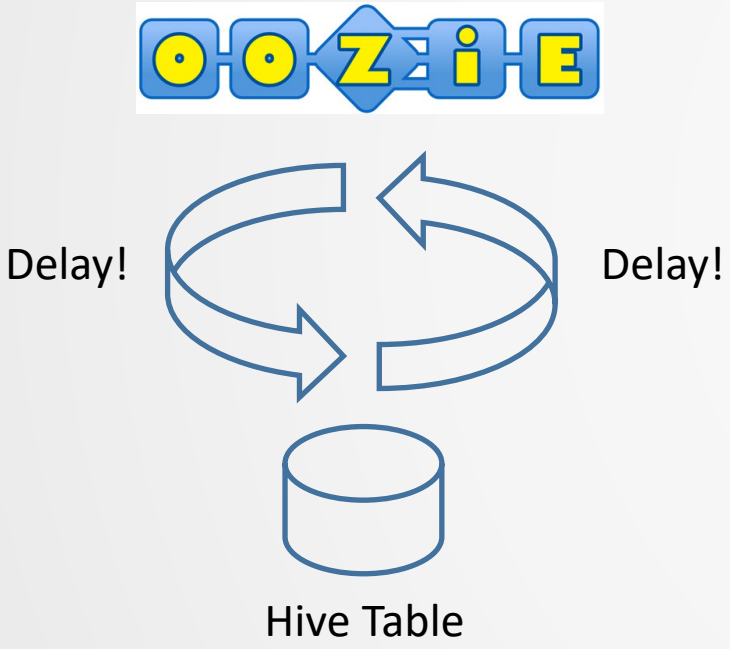


像批一样简单的关联 Hive tables (Broadcast)：

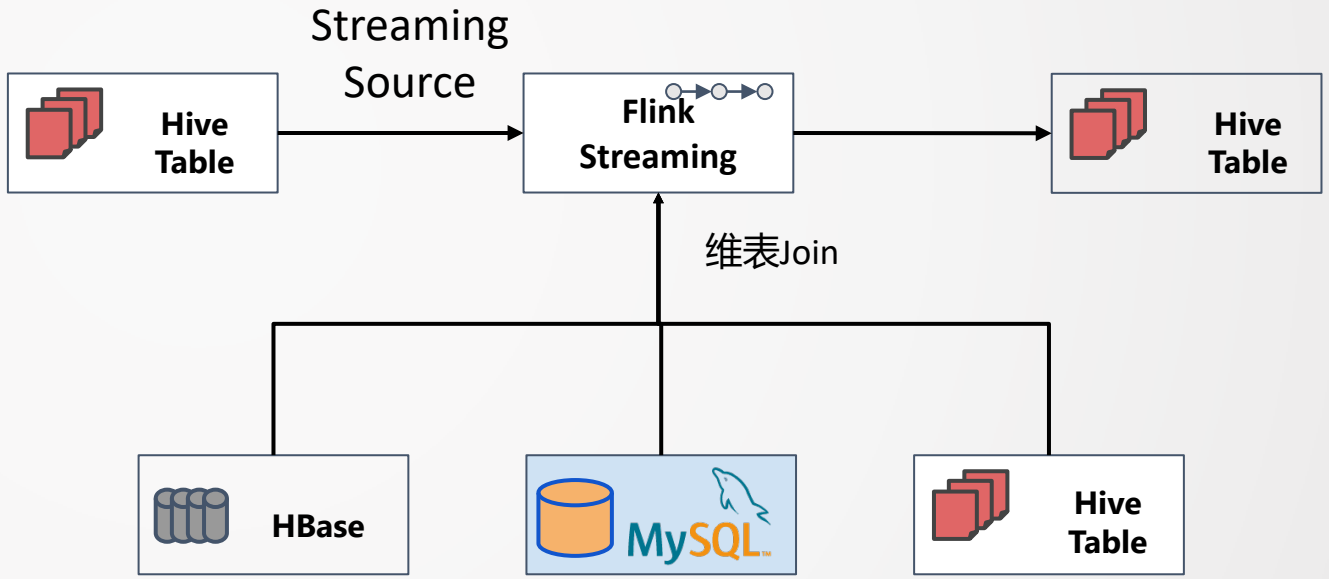


不止准实时摄入：准实时消费

调度工具



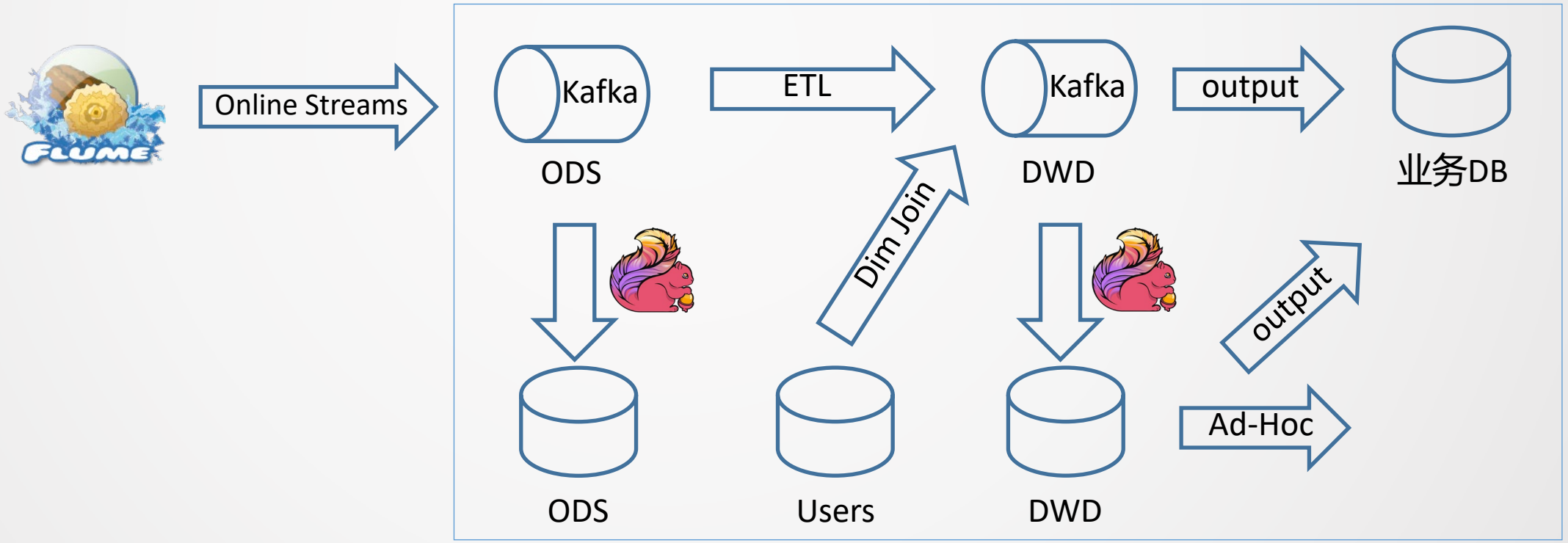
- 实时读取 (分区只消费一次)
- 增量计算



构建流批一体准实时数仓应用实践

案例需求

- 通过打点日志Logs计算各年龄层的PVs
- 实时链路：输入访问日志，关联Hive的User表，计算PVs输出到业务DB
- 离线链路：小时分区表，支持Ad-Hoc查询

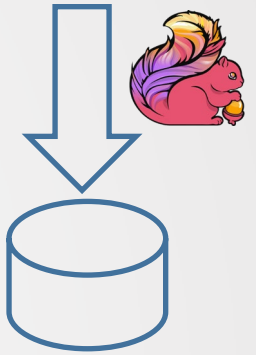


ODS – 数据摄入案例

ODS

- Use catalog hive_catalog;
- Create database realtime_db;
- Create database offline_db;

统一的元数据管理



ODS

- Set table.sql-dialect=hive;
- CREATE TABLE offline_db.click (user_id bigint, ts string)
  PARTITIONED BY (dt STRING, hour STRING)
  STORED AS PARQUET TBLPROPERTIES (
  'sink.partition-commit.trigger'='partition-time',
  'partition.time-extractor.timestamp-pattern'='$dt $hour:00:00',
  'sink.partition-commit.policy.kind'='metastore,success-file'
  );

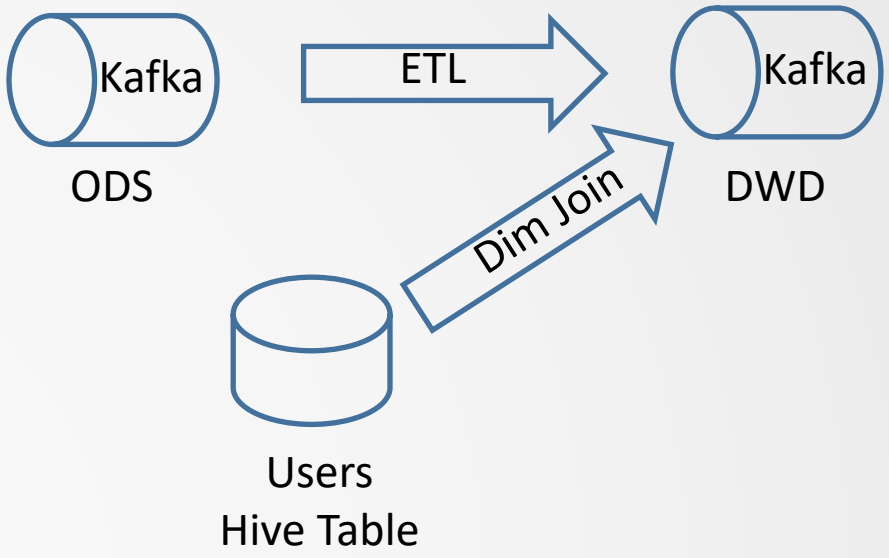
基于Watermark的精确分区提交策略

- SET table.sql-dialect=default;
- Create table realtime_db.click ...;
- INSERT INTO TABLE offline_db.click SELECT
  user_id, ts, DATE_FORMAT(log_ts, 'yyyy-MM-dd'),
  DATE_FORMAT(log_ts, 'HH')
  FROM realtime_db.click;

读取Kafka写入Hive Table

实时和离线的差别在于分区

DWD – 实时维表关联



- Set table.sql-dialect=hive;
- CREATE TABLE offline_db.users (user_id bigint, age int)
  STORED AS PARQUET TBLPROPERTIES (
  'lookup.join.cache.ttl' = '1h' – 小时级更新维表
  );

- SET table.sql-dialect=default;

- CREATE VIEW click_with_time AS SELECT user_id, ts,
  PROCTIME() as proctime FROM realtime_db.click;

Flink Dim Join需要生成proctime

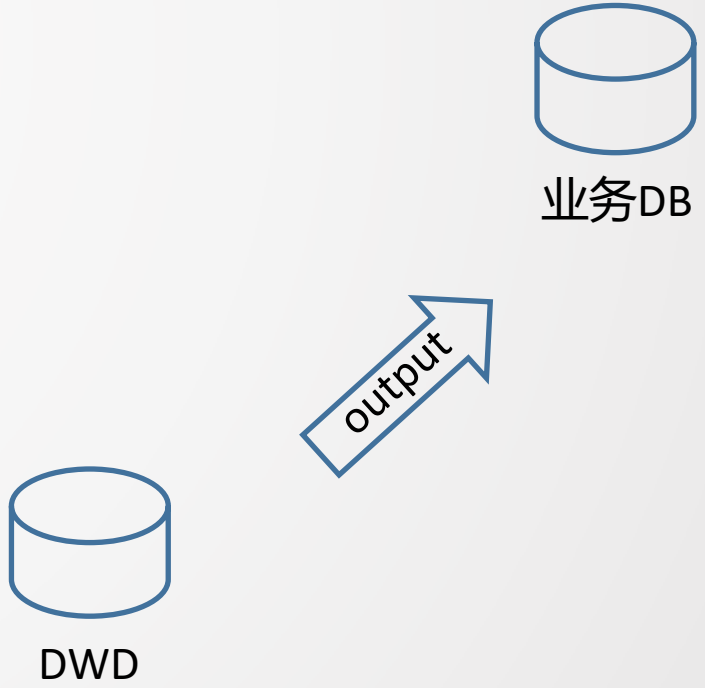
- INSERT INTO TABLE realtime_db.age_click
  SELECT user_id, age, ts FROM realtime_db.click AS T
  JOIN offline_db.users FOR SYSTEM_TIME AS OF T.proctime AS D
  ON T.user_id = D.user_id;

Flink Dim Join语法

OUTPUT – Hive上的实时Pipeline

- INSERT INTO TABLE mysql_db.pv
  SELECT dt, age, COUNT(*) AS cnt FROM offline_db.age_click
  /*+ OPTIONS('streaming-source.enable'='true',
  'streaming-source.consume-start-offset'='2020-05-20') */
  GROUP BY dt, age;

- 以streaming的方式读取Hive表
- 从2020-05-20号开始消费



THANKS!