基于 Apache Flink 的爱奇艺实时计算平台建设实践

Real-time Computing Platform with Apache Flink at iQIYI

梁建煌
Jianhuang Liang

爱奇艺大数据服务负责人
Leader of iQIYI Bigdata Team

FLINK FORWARD # ASIA

实时即未来 # Real-time Is The Future

FLINK FORWARD

基于 Apache Flink 的爱奇艺实时计算平台建设实践

Real-time Computing Platform with Apache Flink at iQIYI

Flink 现状与改进

Flink Usage and Improvements

实时计算平台

Real-time Computing Platform

Flink 业务案例

Flink Use Cases

爱奇艺大数据服务发展历史

Evolution of bigdata service at iQIYI

2012 — **1st Hadoop Cluster**
MapReduce/Hive
20 nodes

2014 — **Storm/Spark**
Hadoop 2.0

2015 — **Europa**
Streaming Job Platform
Kafka

2017 — **Flink**
StreamingSQL

2018 — **RAP**
Real-time Analytics Platform

2019 — **Stream Processing Platform**
15000+ nodes

Flink @ iQIYI

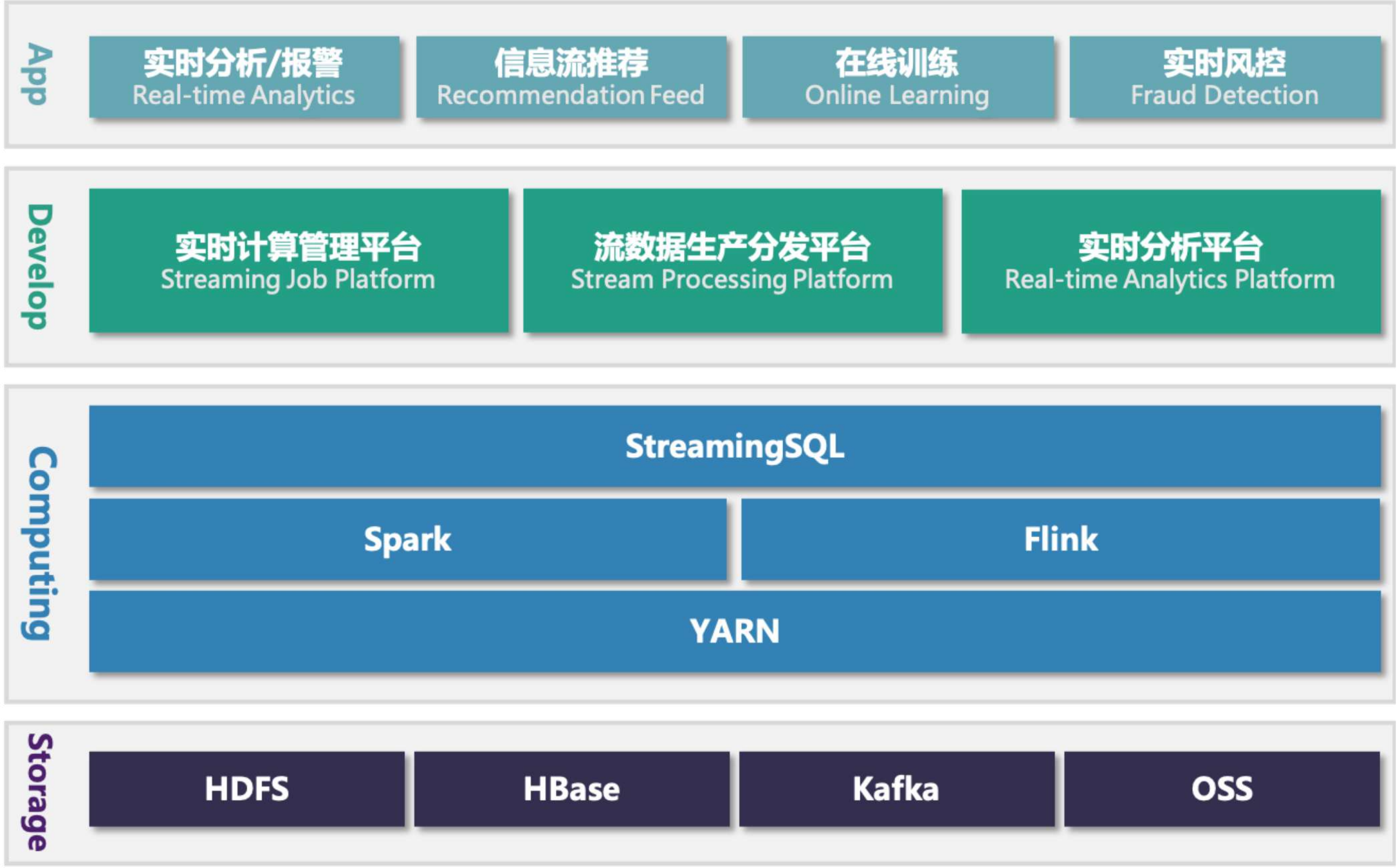
15000+
Nodes

800+
Jobs

1000 Billion
Records/day

2500 TB/day

Flink @ iQIYI



业务赋能
Business empowerment

平台化建设
Platform construction

统一 SQL 引擎
Unified SQL engine

Flink 增强
Flink enhancement

Flink 改进 - 监控和报警

Flink Improvements - Monitoring and Alerting

Flink 作业监控数据和报警集成爱奇艺 Hubble 统一监控平台
Flink job metrics and alerts are integrated into iQIYI Hubble monitoring system

Job 级别监控指标：Job 状态、Checkpoint 状态和耗时
Job-level metrics: job status, checkpoint status and elapsed time

Operator 级别监控指标：时延、反压、Source/Sink 流量，对每个 Operator 进行指标聚合
Operator-level metrics: latency, backpressure, source and sink traffic, aggregation for each operator

TaskManager 级别监控指标：CPU 使用率、内存使用率、JVM GC 等
TaskManager-level metrics: CPU usage, memory usage, JVM GC, etc.

Flink 改进 - 状态管理

Flink Improvements - State Management

问题一：Checkpoint 只在 Flink 作业内部有效，主动重启或异常重启时，如何从上次运行的状态恢复？
Problem 1: A checkpoint is internal to a Flink job. How to restore from the previous state when a job is restarted manually or accidentally?

解决方法：作业重启时，找到上一次成功的 Checkpoint，从中恢复
Solution: Find the last successful checkpoint and restore from it.

缺陷：对于状态很大的的作业，使用 RocksDBStateBackend 做增量 Checkpoint；上一次 Checkpoint 被依赖而无法删除，会导致状态堆积（生产环境中一个作业的 Checkpoint 总共多达 8 TB）
Drawback: For jobs with large state, incremental checkpoints with RocksDBStateBackend are used. Since new checkpoints depend on old ones, checkpoint size will grow indefinitely.

Flink 改进 - 状态管理

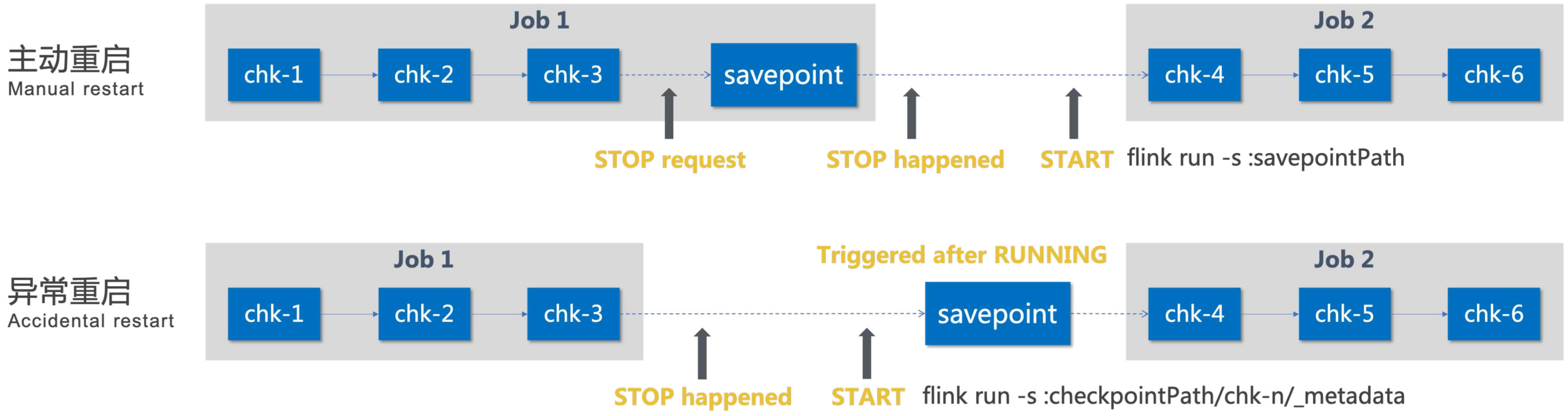
Flink Improvements - State Management

问题二：Checkpoint 无限依赖

Problem 2: Dependency chain of checkpoints

解决方法：使用 Savepoint 打断增量 Checkpoint 的依赖链，并与流计算平台集成

Solution: Use Savepoint to break the dependency chain of checkpoints, taken care of by our streaming job platform



StreamingSQL

基于 Spark、Flink 构建的流数据 ETL 工具
Streaming ETL tool based on Spark and Flink

SQL 化：只需要通过编写 SQL 即可完成流计算 ETL 任务的开发
SQL interface: easy to develop streaming ETL jobs with SQL

DDL：流表、临时表、维度表、结果表
DDL: Stream Table, Temporary Table, Dimension Table, Result Table

UDF：系统预定义常用函数、用户自定义函数
UDF: frequently used built-in functions and user-defined functions

提供 SQL 编辑器
SQL Editor provided

StreamingSQL Example

```
CREATE STREAM TABLE t1 (ts long, a string, b int, c double) WITH (
  type="kafka",
  brokers = "ip1:9092,ip2:9092,ip3:9092",
  topics = "some_topic",
  deserializer = "json.weak");

CREATE TMP TABLE t2 as
  select
    cast(floor(ts / 300000) * 300 as timestamp) as d_dt,
    a, b, c from t1;

CREATE tmp table t3 as
  select d_dt, a, max(b) as max_b, sum(c) as sum_c, count(c) as cnt_c from t2;

create result table r (d_dt timestamp, a varchar(255), max_b int, sum_c double, cnt_c bigint)
  with (
    type="mysql",
    url="jdbc:mysql://ip:port/schema",
    table="some_table",
    username="some_username",
    password="some_password",
    query="insert into some_table (d_dt, a, max_b, sum_c, cnt_c) values (?,?,?,?,?)
           ON duplicate key update
              max_b = IF(?<values(max_b),values(max_b) , ?),
              sum_c=values(sum_c)+?, cnt_c=values(cnt_c)+?",
    fields=[d_dt, a, max_b, sum_c, cnt_c, max_b, max_b, sum_c, cnt_c]
  );

upsert into r
select * from t2;
```

流表：实时流输入及反序列化方式，目前支持 Kafka

Stream Table: define streaming input and deserializer, only support Kafka now

维度表：静态表，用于与流表 join

Dimension Table: static table joining with Stream Table

临时表：中间结果

Temporary Table: intermediate result

结果表：输出数据源，支持 MySQL、Kafka、ES、HBase、Druid、Kudu

Result Table: data sink, supporting MySQL, Kafka, ES, HBase, Druid, Kudu, etc.

基于 Apache Flink 的爱奇艺实时计算平台建设实践

Real-time Computing Platform with Apache Flink at iQIYI

Flink 现状与改进

Flink Usage and Improvements

实时计算平台

Real-time Computing Platform

Flink 业务案例

Flink Use Cases

实时计算管理平台

Streaming Task Management Platform

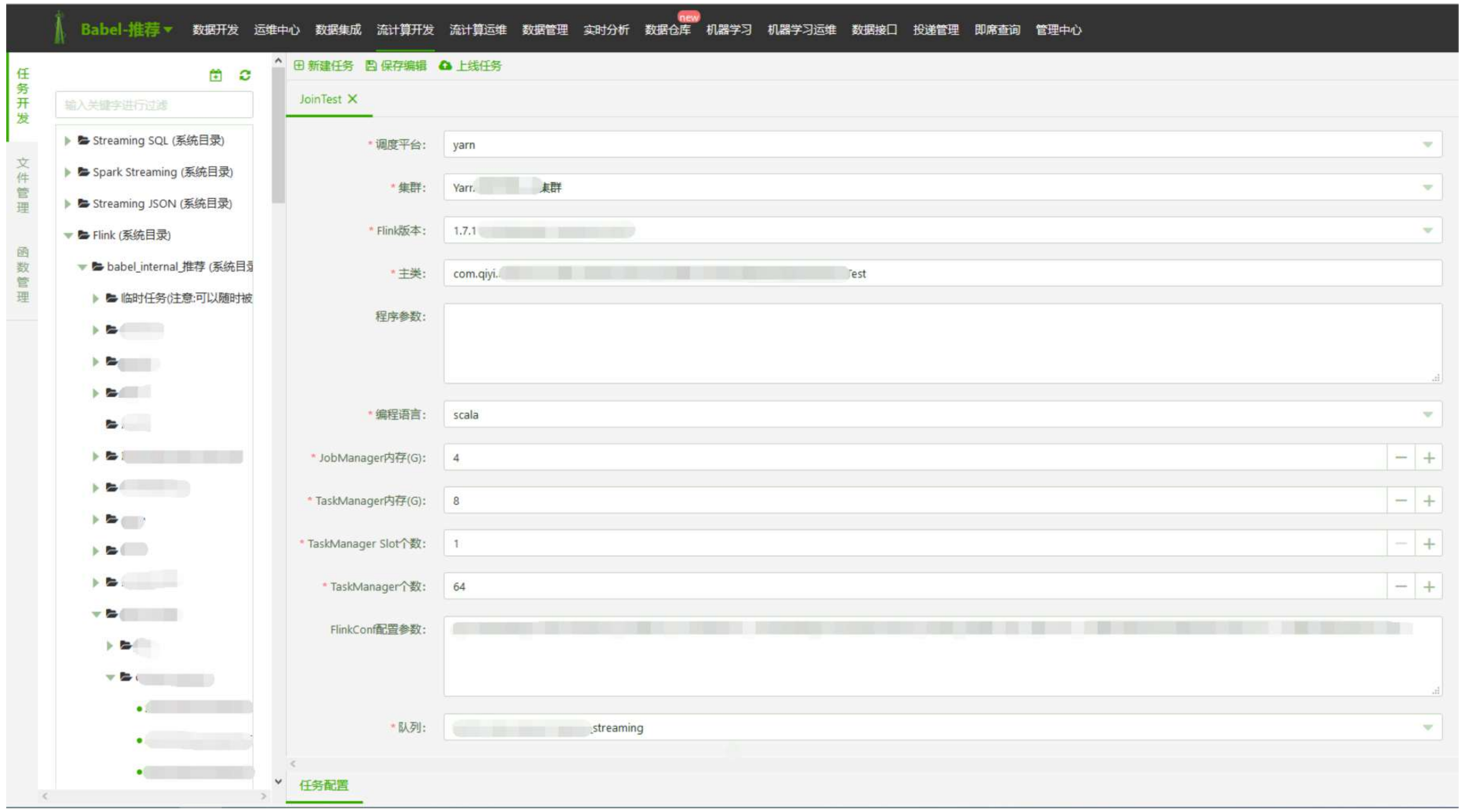
Spark、Flink 任务开发和管理 web IDE
Development and management web IDE for Spark and Flink jobs

文件管理：任务 Jar 包、依赖库
File management: job's jar and referenced libraries

函数管理：提供丰富的系统函数、支持用户注册 UDF
UDF Management: system built-in functions and user-defined functions

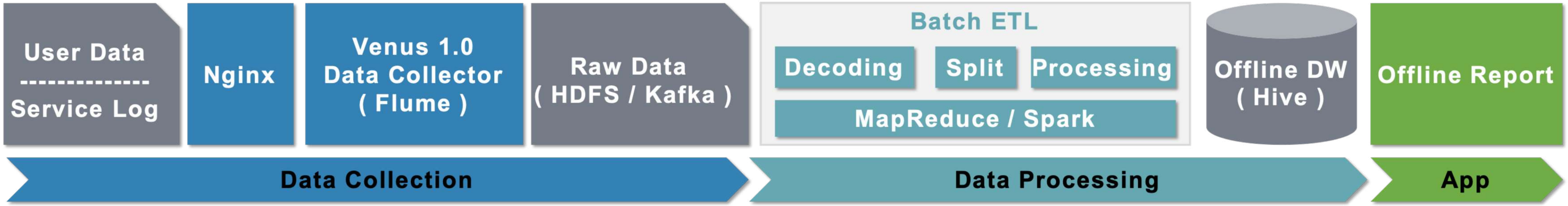
版本管理：支持任务、文件的版本对比及回滚
Version Control: version comparison and rollback of jobs and files

监控大盘、报警订阅、资源审计、异常诊断
Monitoring Dashboard, Alerts Subscription, Resource Audit, Task Diagnosis



实时数据处理平台演进 (2015 - 2016)

Evolution of Stream Processing System (2015 - 2016)



场景：离线报表为主，少量实时报表需求，数据生产规模 50 万 QPS

Scenario: mainly offline data reports, only few near real-time reports, 500,000 QPS

Venus 1.0 数据采集平台

Venus 1.0 Collector

- 基于 Apache Flume
  Based on Apache Flume
- 在 Venus agents 上通过 tail + grep/awk/sed 等脚本过滤
  Filtering data by tail, grep, awk or sed shell commands on Venus agents

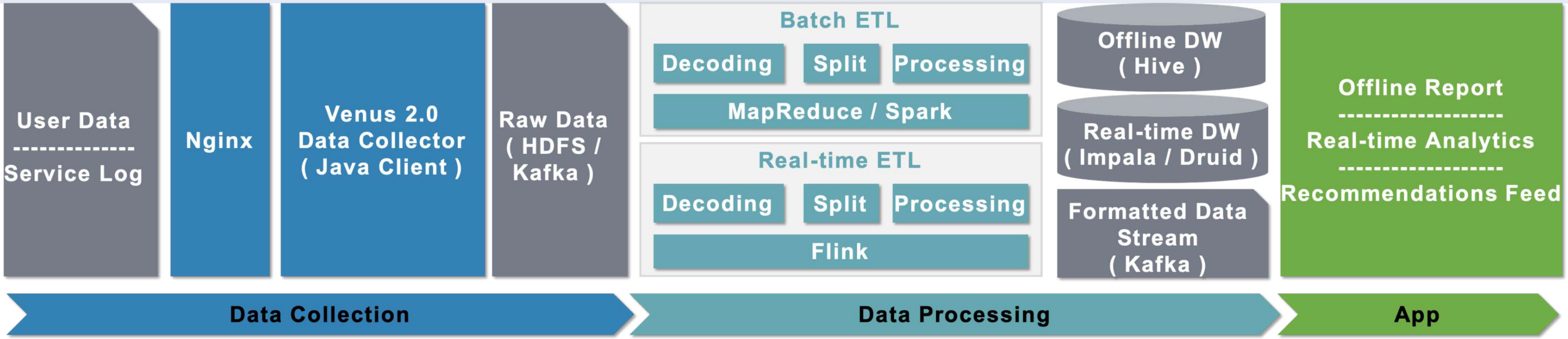
缺陷

Drawbacks

- 不方便变更过滤规则，需重启所有 agents
  Inconvenient to update filters as need to restart all agents
- 不同用户需求存在大量重复处理逻辑
  Duplicated processing logic among different users

实时数据处理平台演进 (2017 - 2018)

Evolution of Stream Processing System (2017 - 2018)



场景：实时分析、信息流推荐等实时需求增加，500 万 QPS

Scenario: increased demand for real-time analytics, recommendations feed, etc., 5 Million QPS

Venus 2.0 数据采集分析平台

Venus 2.0 Data Processing Platform

- 实时过滤从 Venus agent 迁移到 Flink，采用两级 Kafka
  Move real-time filters from Venus agent to Flink by using two-stage Kafka clusters
- 无需重启即可动态增减处理规则
  Dynamically update processing rules without restarting Venus agents or Flink jobs

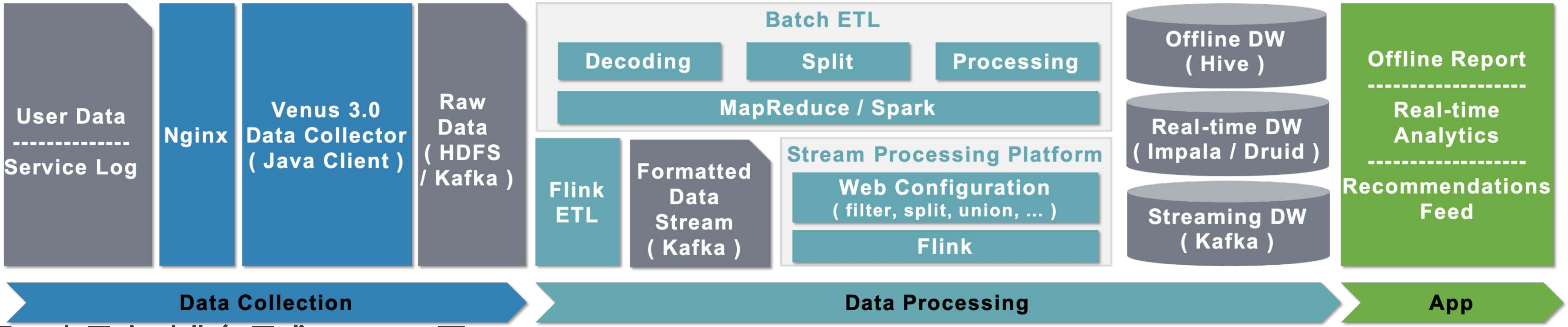
缺陷

Drawbacks

- Kafka 数据冗余
  Duplication of Kafka data stream
- 不方便分享 Kafka 数据
  Inconvenient to share Kafka data

实时数据处理平台演进（2019）

Evolution of Stream Processing System (2019)



场景：大量实时业务需求，1500 万 QPS

Scenario: strong demand for real-time computing, 15 Million QPS

Venus 3.0 流数据生产分发平台

Venus 3.0 Stream Processing Platform

- 通过 web 配置实时处理规则，可自由组合常见算子
  Configure stream data processing rules on web with free combination of commonly-used operators
- 参考离线数仓，按照数据使用场景构建流式数仓
  Build streaming data warehouse based on scenarios

优点

Pros

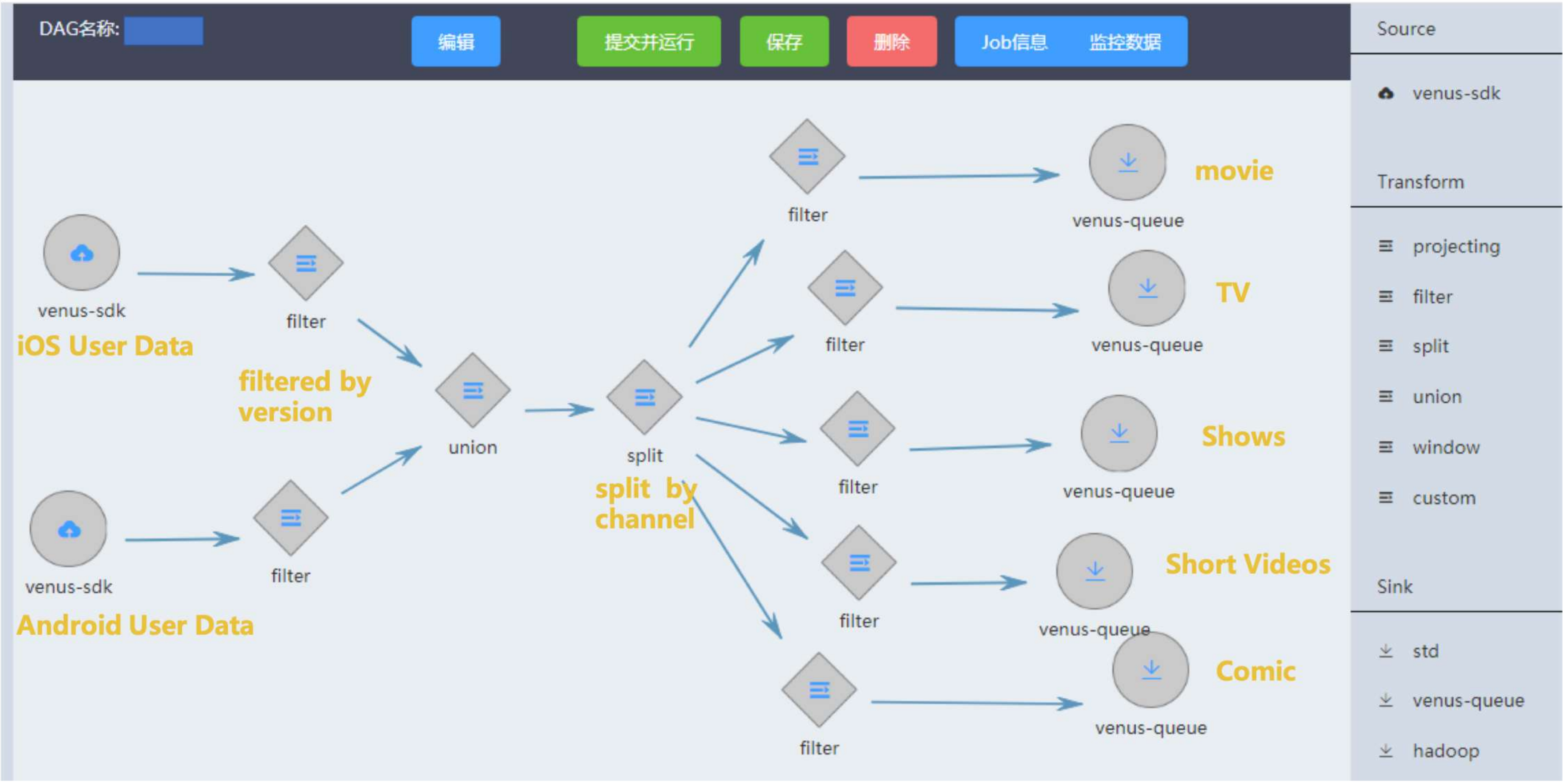
- 减少流数据重复生产
  Reduce stream data duplication
- 促进流数据共享
  Facilitate stream data sharing among different projects

实时数据处理平台演进（2019）

Evolution of Stream Processing System (2019)

支持 Projection、Filter、Split、Union、Window、UDF 等常见算子

Support commonly-used stream operators like Projection, Filter, Split, Union, Window, UDF, etc.



实时分析平台

Real-time Analytics Platform

海量实时数据 OLAP 分析平台

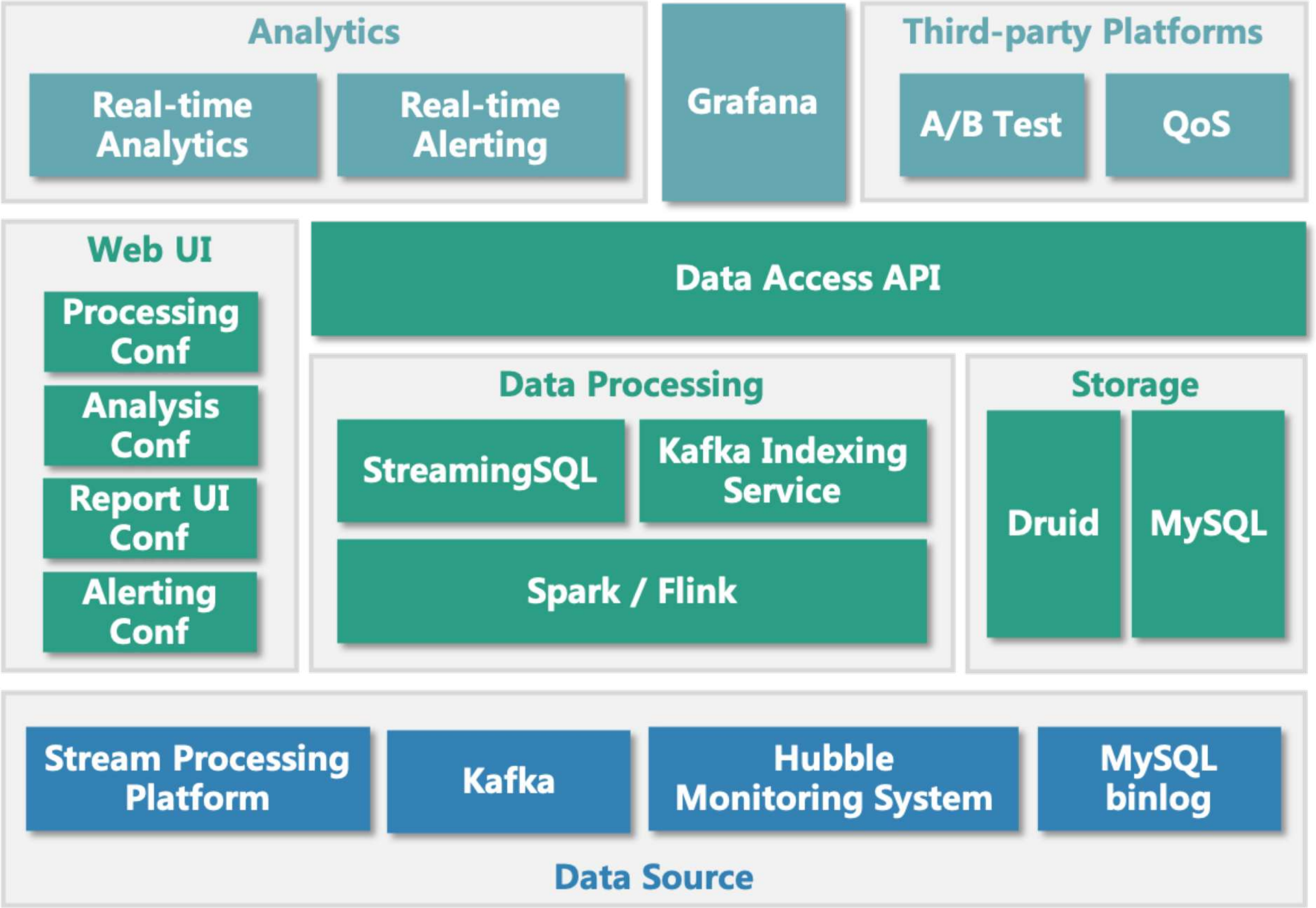
Real-time big data analytics platform

- **实时报表：A/B 测试、精细化运营等**
  Real-time reports for A/B test, precision marketing, etc.
- **实时报警：VV/UV、播放故障等**
  Real-time monitoring and alerting

优势

Pros

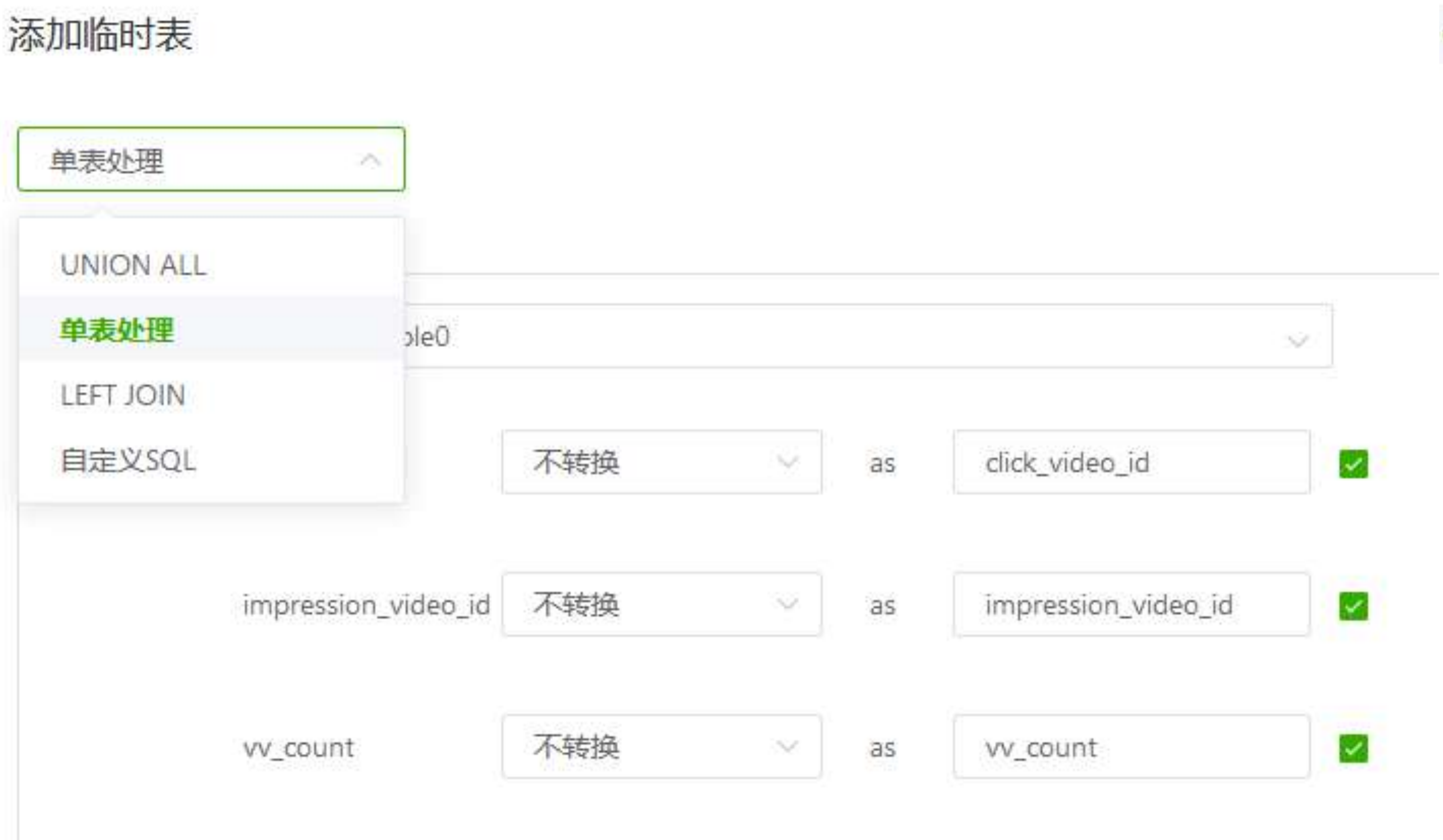
- **开发门槛低：无需写程序或 SQL**
  Easy analysis: no need to code with Scala or SQL
- **开发效率高：几天 -> 半小时**
  High efficiency: days -> 30 mins
- **报表实时：小时 -> 1 分钟**
  Low data latency: hours -> 1 min
- **查询更快：支持大规模数据亚秒级查询**
  Fast query: sub-seconds query response with large-scale data



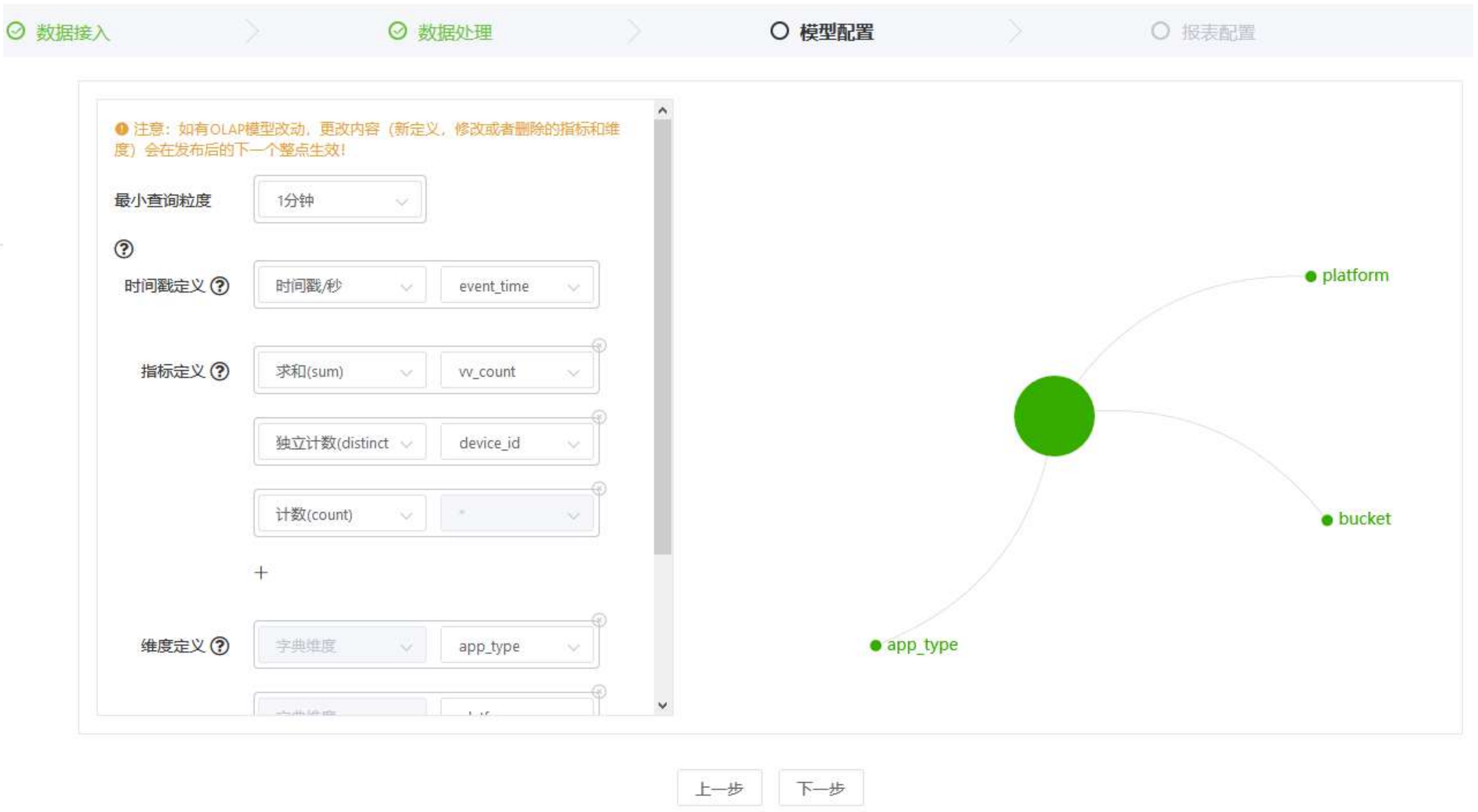
实时分析平台

Real-time Analytics Platform

配置处理规则
Configure processing rules



配置 OLAP 模型
Configure OLAP model



基于 Apache Flink 的爱奇艺实时计算平台建设实践

Real-time Computing Platform with Apache Flink at iQIYI

Flink 现状与改进

Flink Usage and Improvements

实时计算平台

Real-time Computing Platform

Flink 业务案例

Flink Use Cases

信息流推荐

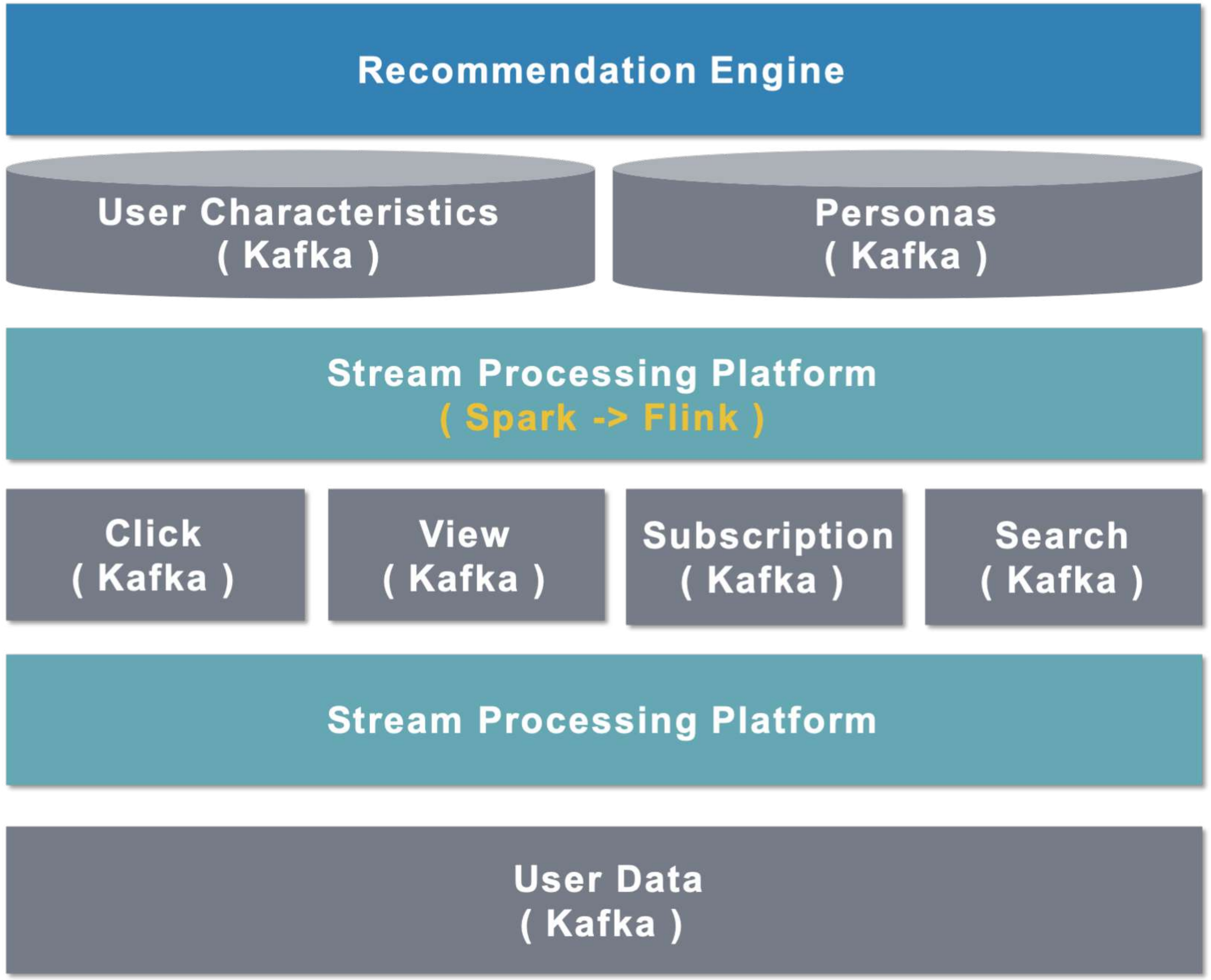
Recommendations Feed

从 Spark Streaming 迁移到 Flink，消除批处理延迟
Move from Spark Streaming to Flink to eliminate batch delay

单个任务延迟从 1 分钟缩短到 1 - 2 秒
Job latency reduced from 60 seconds to 1 to 2 seconds

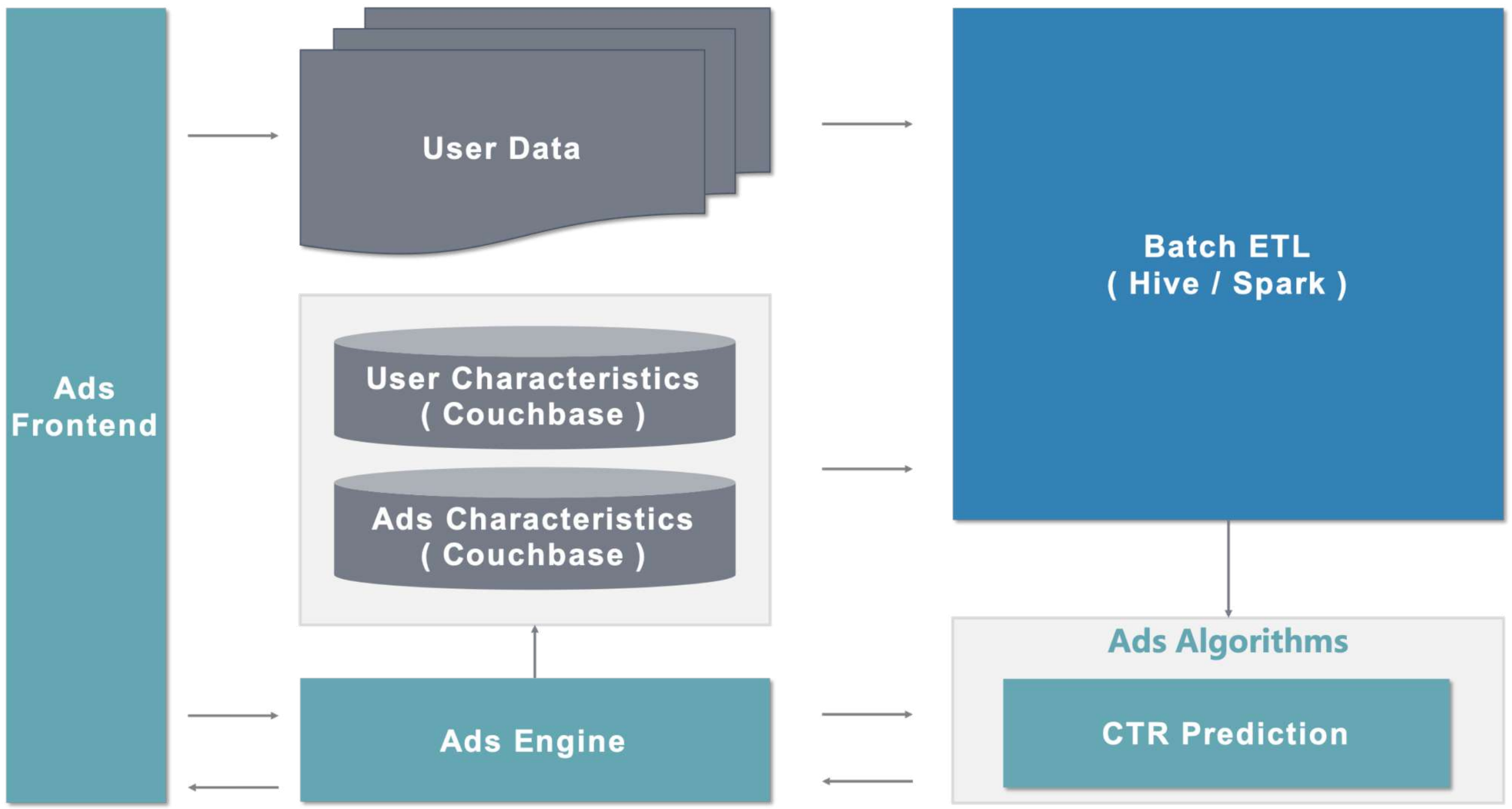
端到端性能提升 86 倍
End-to-end performance is improved by 86X

显著提升推荐效果
As a result, recommendation effect is improved significantly



使用 Flink 生产深度学习训练数据

Real-time Processing with Flink for Deep Learning



通过 Hive/Spark 离线 ETL 生成广告深度学习算法所需训练数据

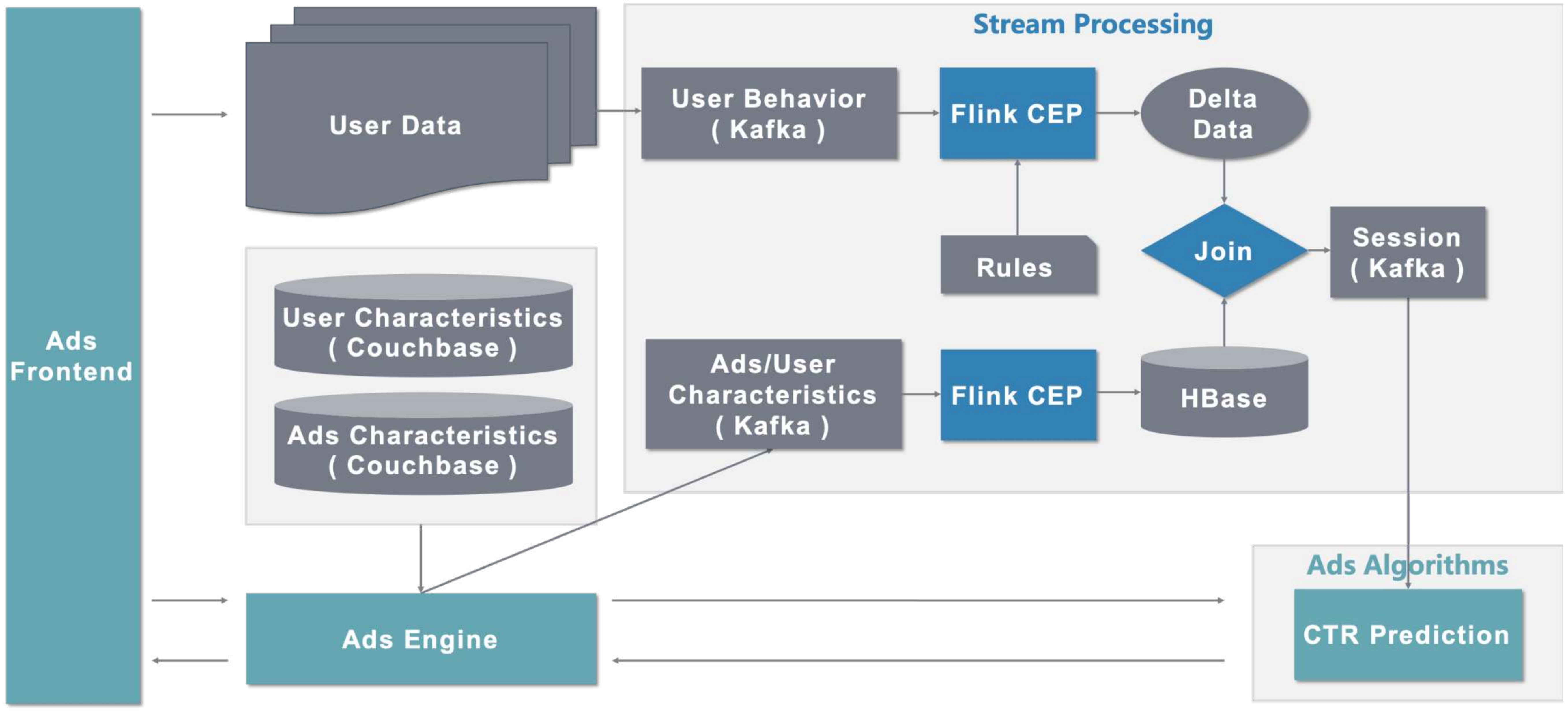
Training data for Ads deep learning algorithms were generated through batch ETL with Hive and Spark

算法模型更新周期为 6 小时

The iteration period of model was 6 hours

使用 Flink 生产深度学习训练数据

Real-time Processing with Flink for Deep Learning



Kafka 实时流（最近 24 小时）join HBase 维度表（最近 7 天）

Kafka stream (last 24 hours data) join with HBase table (last 7 days data)

算法模型更新从 6 小时缩短到 1 小时

The iteration period of model reduced from 6 hours to 1 hour

支持实时 CTR 预估，更好指导广告决策

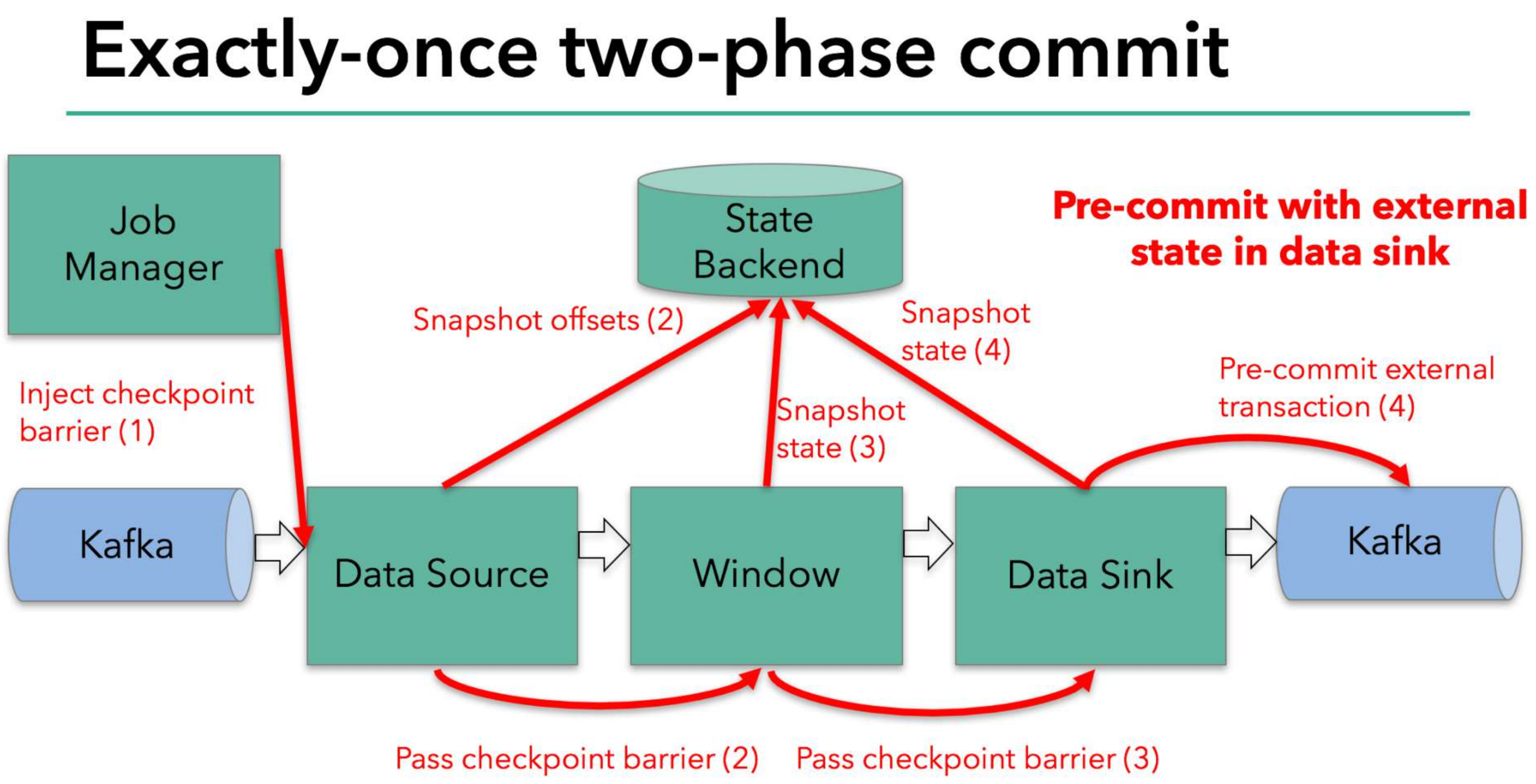
Support real-time CTR estimation to make better advertising decisions

效果：提升广告收益

Effect: increase advertising revenue

端到端 Exactly-Once 处理

End-to-End Exactly-Once Processing



问题：Kafka 节点故障重启或者人工运维时，业务方重复消费数据

Problem: One Kafka node failure can cause duplicated data consumption

方案：Kafka Exactly Once Semantics + Flink two-phase commit

Solution: Kafka Exactly Once Semantics + Flink two-phase commit

Flink 任务计算性能损耗 20%

20% loss of Flink task performance

挑战与规划

Challenges and Future Work

流批一体化
Unified batch and real-time stream processing

SQL 化：进一步完善和推广 StreamingSQL，降低开发门槛
Improve and promote StreamingSQL, make ETL and data analytics much easier

基于 Flink 的机器学习
Machine learning with Flink

提高 Flink 作业的资源利用率，支持动态资源调整
Increase resource utilization of Flink jobs and enable dynamic resource allocation

Flink on Kubernetes

FLINK FORWARD

THANKS