FLINK FORWARD # ASIA 2022 ONLINE | 实时即未来

Apache Flink在翼支付的实践应用

尹春光 | 翼支付 高级大数据工程师

01 公司实时业务场景

公司简介

天翼电子商务有限公司（以下简称“翼支付”）是中国电信集团有限公司的成员企业，是国资委双百改革和发改委第四批混改“双试点”企业，也是“双试点”企业中唯一的金融科技公司。公司以翼支付APP为载体，面向7000万月活用户，提供民生缴费、消费购物、金融理财等服务内容，依托区块链、云计算、大数据、人工智能等技术，赋能超1000万家线下商户门店及170余家线上知名电商。

秉持“响应监管、服务民生、资源共享、合作多赢”的理念，聚焦“开放、安全、便捷”的核心产品力，翼支付坚持通过服务投入与产品升级，构建贴合需求的管理与业务体系，以交流融合的业务实践，推动产业各方实现数字化转型。

业务挑战

金融大数据场景：

1、海量数据处理

2、高并发请求

3、低延迟时效性

4、业务多样性

5、场景复杂性

02 平台简介

翼支付流计算发展历程介绍

阶段一

SparkStreaming应用研发

阶段二

SparkStructStreaing实时平台开发

阶段三

基于FlinkSQL构建StreamingSQL引擎

阶段四

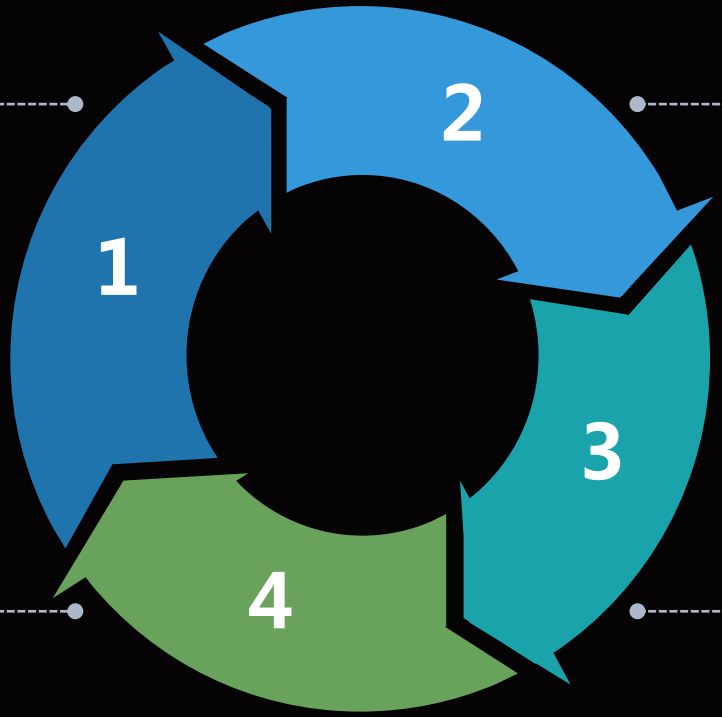
探索基于FlinkCDC+Hudi构建湖仓一体

满足多场景实时任务开发

单击此处添加文本具体内容，简明扼要的阐述您的观点。

SQL任务

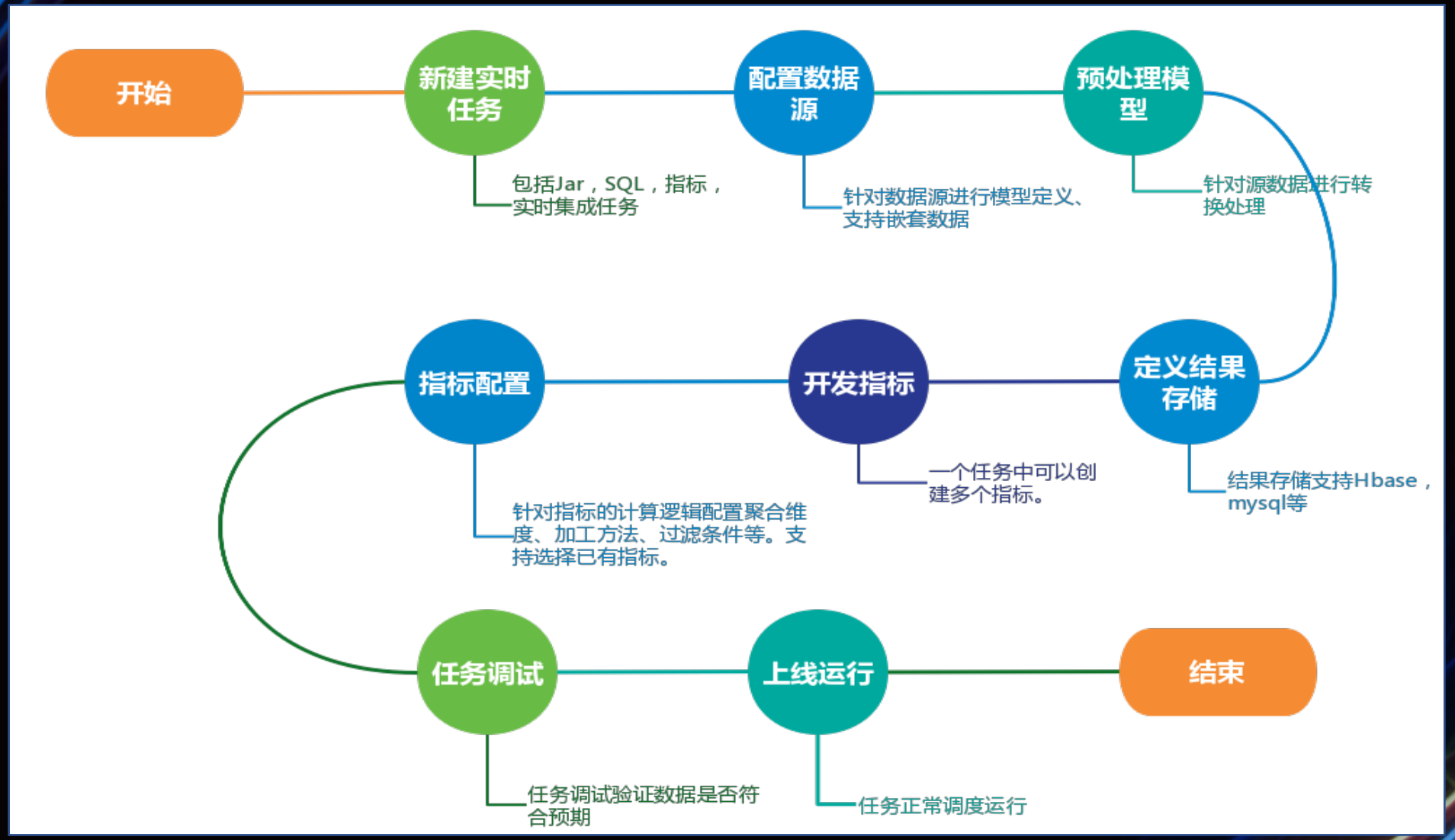
Jar任务



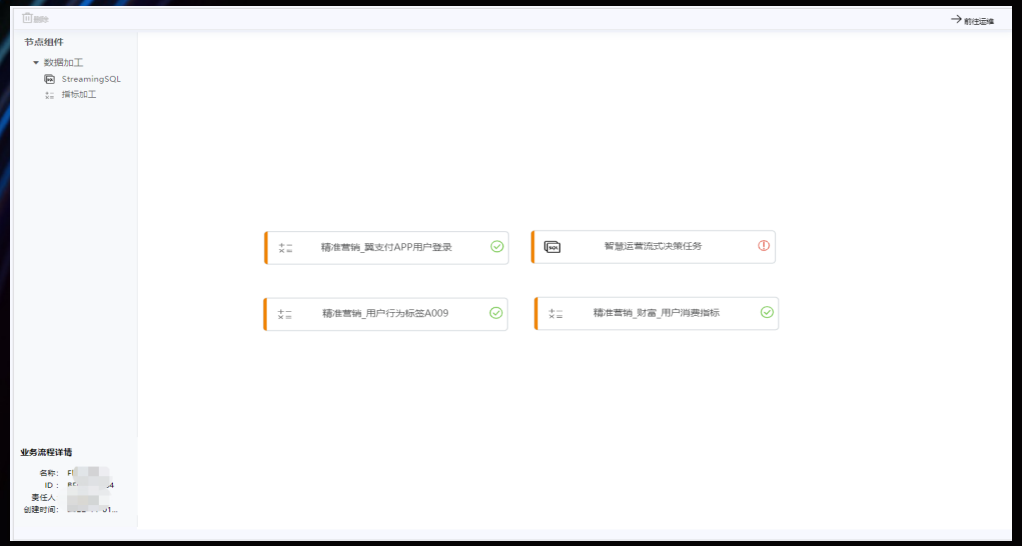
实时特征

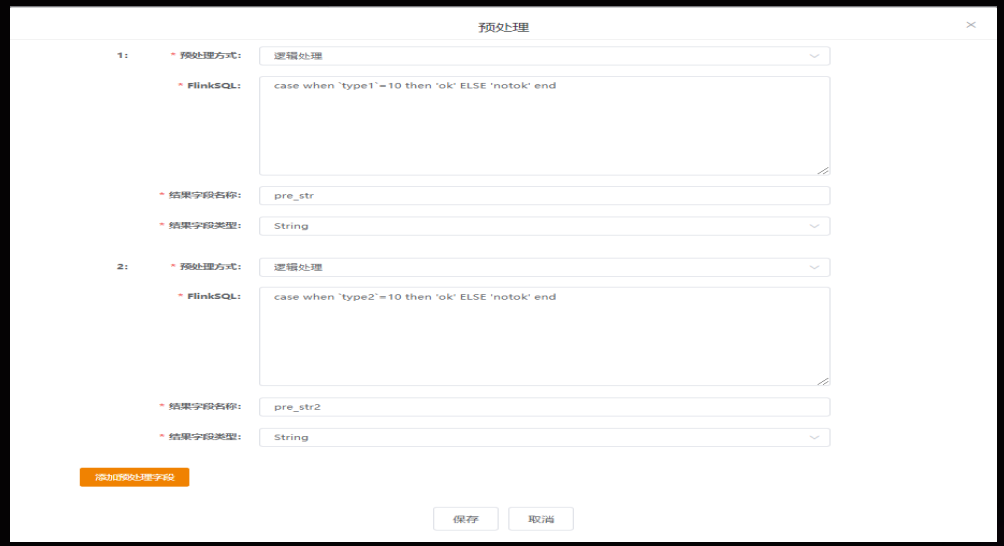
实时数据集成

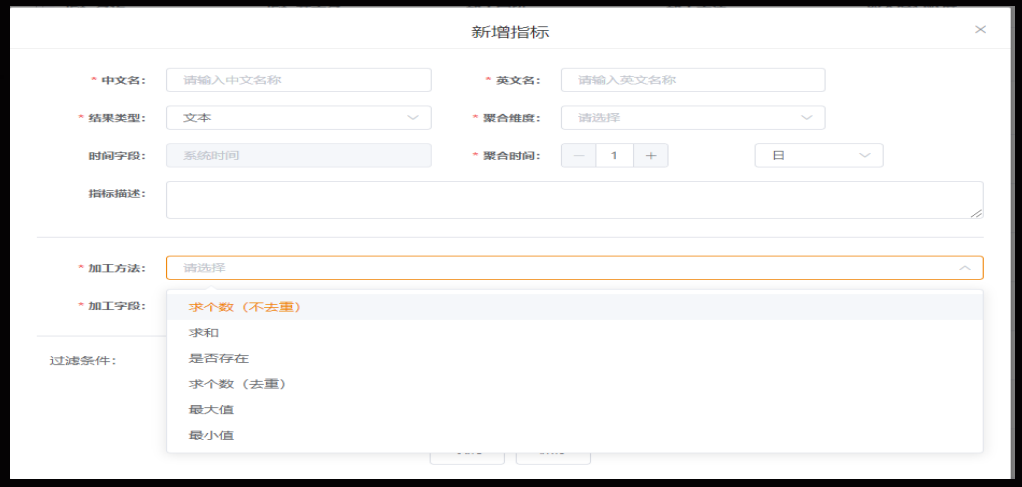
开发流程与任务管理


开始
新建实时任务
包括Jar，SQL，指标，实时集成任务
配置数据源
针对数据源进行模型定义、支持嵌套数据
预处理模型
针对源数据进行转换处理
定义结果存储
结果存储支持Hbase，mysql等
开发指标
一个任务中可以创建多个指标。
指标配置
针对指标的计算逻辑配置聚合维度、加工方法、过滤条件等。支持选择已有指标。
上线运行
任务正常调度运行
任务调试
任务调试验证数据是否符合预期
结束

指标开发流程


节点组件
数据加工
StreamingSQL
指标加工
精准营销_翼支付APP用户登录
智慧运营流式决策任务
精准营销_用户行为标签A009
精准营销_财富_用户消费指标
业务流程详情

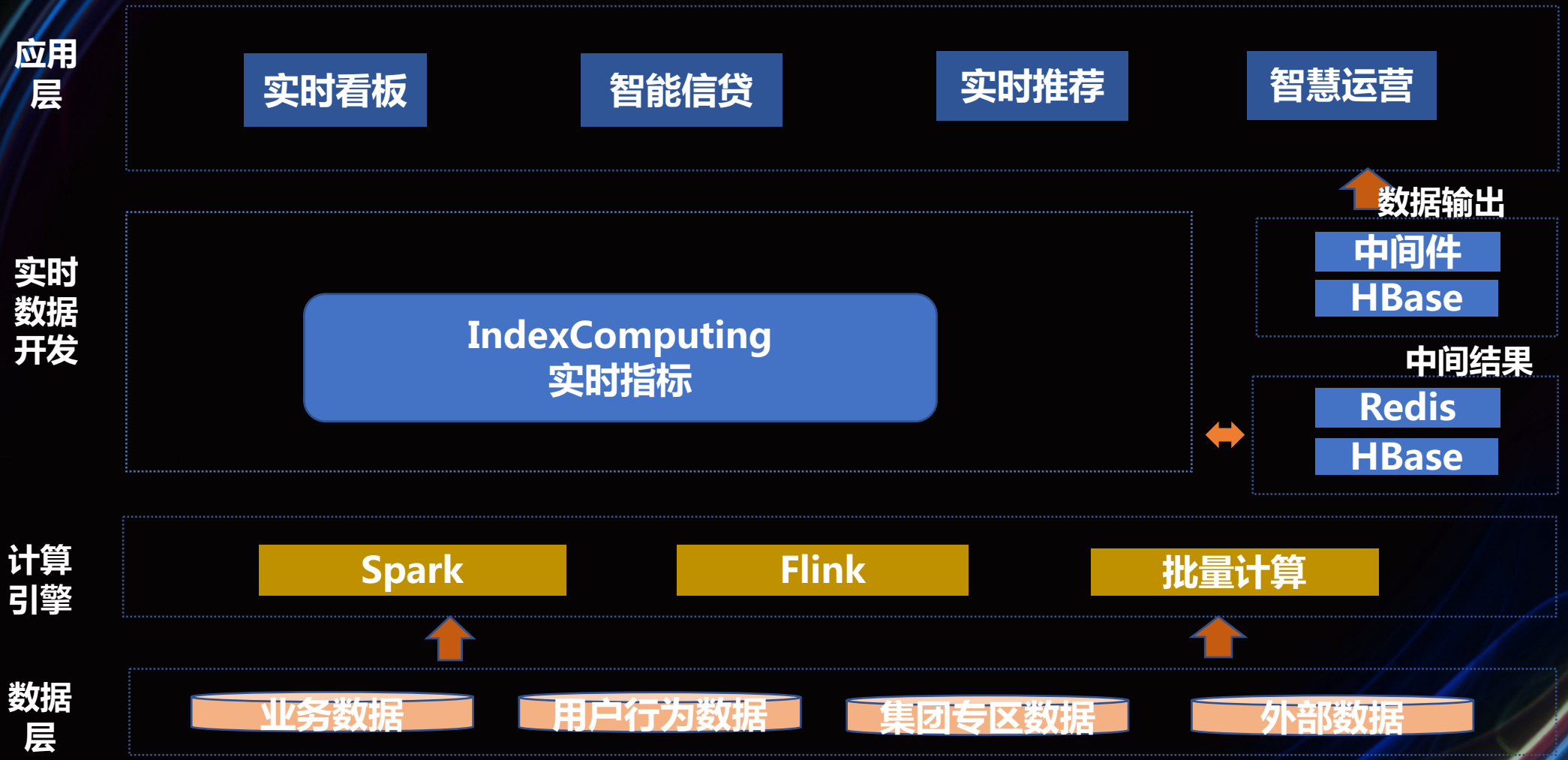

预处理
1：
* 预处理方式：
逻辑处理
* FlinkSQL：
case when `type1`=10 then 'ok' ELSE 'notok' end
* 结果字段名称：
pre_str
* 结果字段类型：
String
2：
* 预处理方式：
逻辑处理
* FlinkSQL：
case when `type2`=10 then 'ok' ELSE 'notok' end
* 结果字段名称：
pre_str2
* 结果字段类型：
String
添加预处理字段
保存
取消


新增指标
* 中文名：
请输入中文名称
* 英文名：
请输入英文名称
* 结果类型：
文本
* 聚合维度：
请选择
时间字段：
系统时间
* 聚合时间：
1
日
指标描述：
* 加工方法：
请选择
* 加工字段：
过滤条件：
求个数（不去重）
求和
是否存在
求个数（去重）
最大值
最小值


回退
指标名称
请输入关键字
新增指标
指标名称
指标英文名
加工字段
加工方法
聚合时间粒度
状态
操作
负数求和_四舍五入_文本_0030
aaa_0030
price003
求和
10日
未生效
详情 | 修改 | 删除
负数求和_过滤条件为负数_向下取整_0030
bbb_0030
price002
求和
4日
未生效
详情 | 修改 | 删除
求个数_去重_文本_0030
ccc_0030
jdp
求个数（不去重）
10分
未生效
详情 | 修改 | 删除
求个数_不去重_文本_0030
ddd_0030
jdp
求个数（不去重）
1时
未生效
详情 | 修改 | 删除
是否存在_不存在_0030
eee_0030
qwer
是否存在
1分
未生效
详情 | 修改 | 删除
是否存在_存在_0030
fff_0030
type1
是否存在
1日
未生效
详情 | 修改 | 删除
预处理_0030
ggg_0030
pre_str
求个数（去重）
1时
未生效
详情 | 修改 | 删除
最大值_0030
max_0030
jdp
最大值
4分
未生效
详情 | 修改 | 删除
最小值_0030
min_0030
jdp
最小值
3时
未生效
详情 | 修改 | 删除
数据源配置
预处理
资源配置
共 9 条
10条/页
前往 1 页

03 平台架构实践

平台架构V1


应用层
实时看板
智能信贷
实时推荐
智慧运营
数据输出
中间件
HBase
实时数据开发
IndexComputing
实时指标
中间结果
Redis
HBase
计算引擎
Spark
Flink
批量计算
数据层
业务数据
用户行为数据
集团专区数据
外部数据

开发引擎模块V1

实时任务开发

任务配置

权限认证

资源配置

任务监控

数据源

中间件

维表

MySql
HBase
Redis

IndexComputing

IndexGroup

DSLParser → Algorithm

数据输出

中间件

HBase

中间结果

Redis

HBase

计算集群

Yarn

HDFS

开发引擎痛点

痛点：

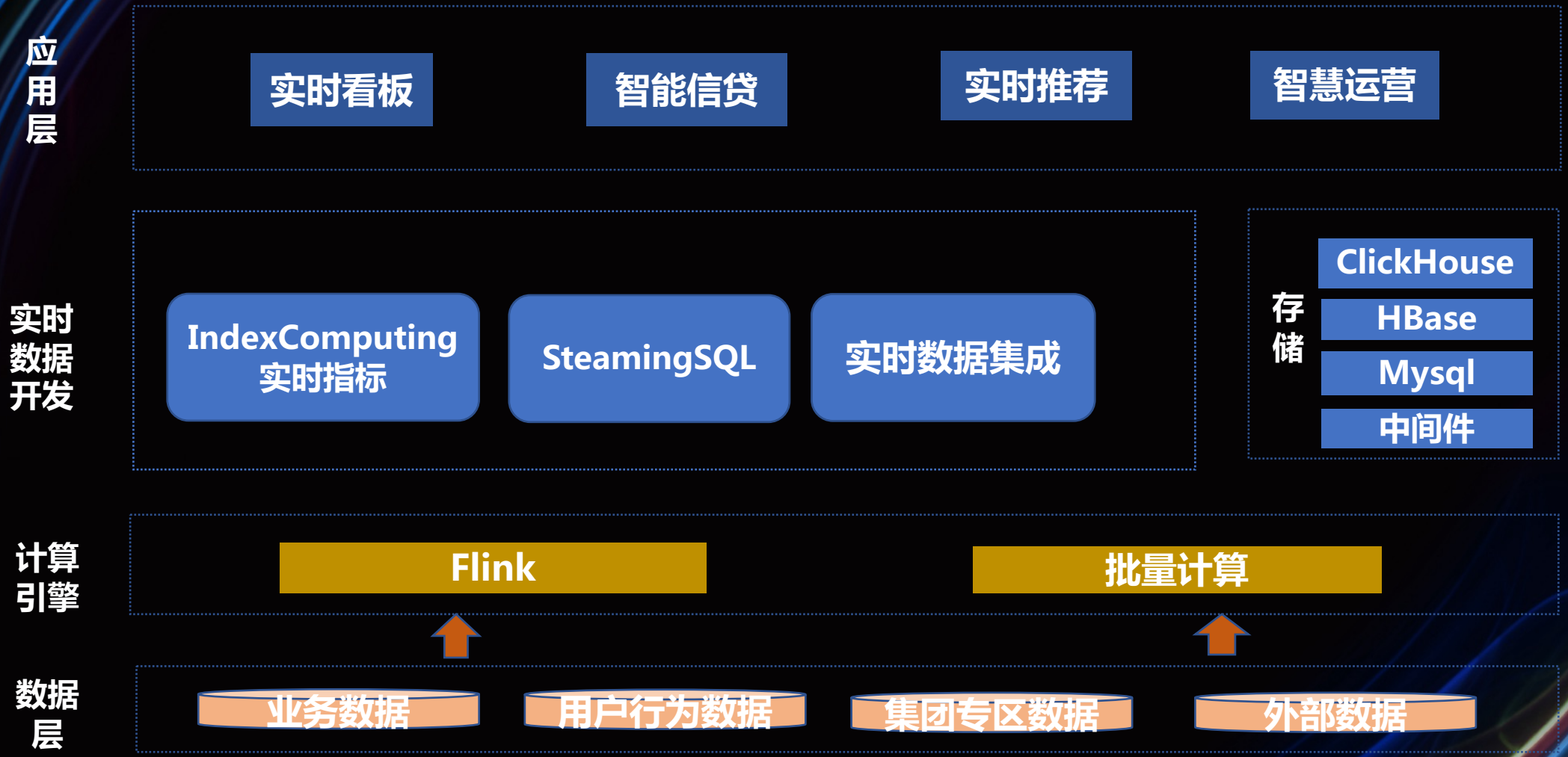
1、需要不断开发计算函数

2、中间结果缓存Redis，中间缓存结果数据量大，聚合计算性能低

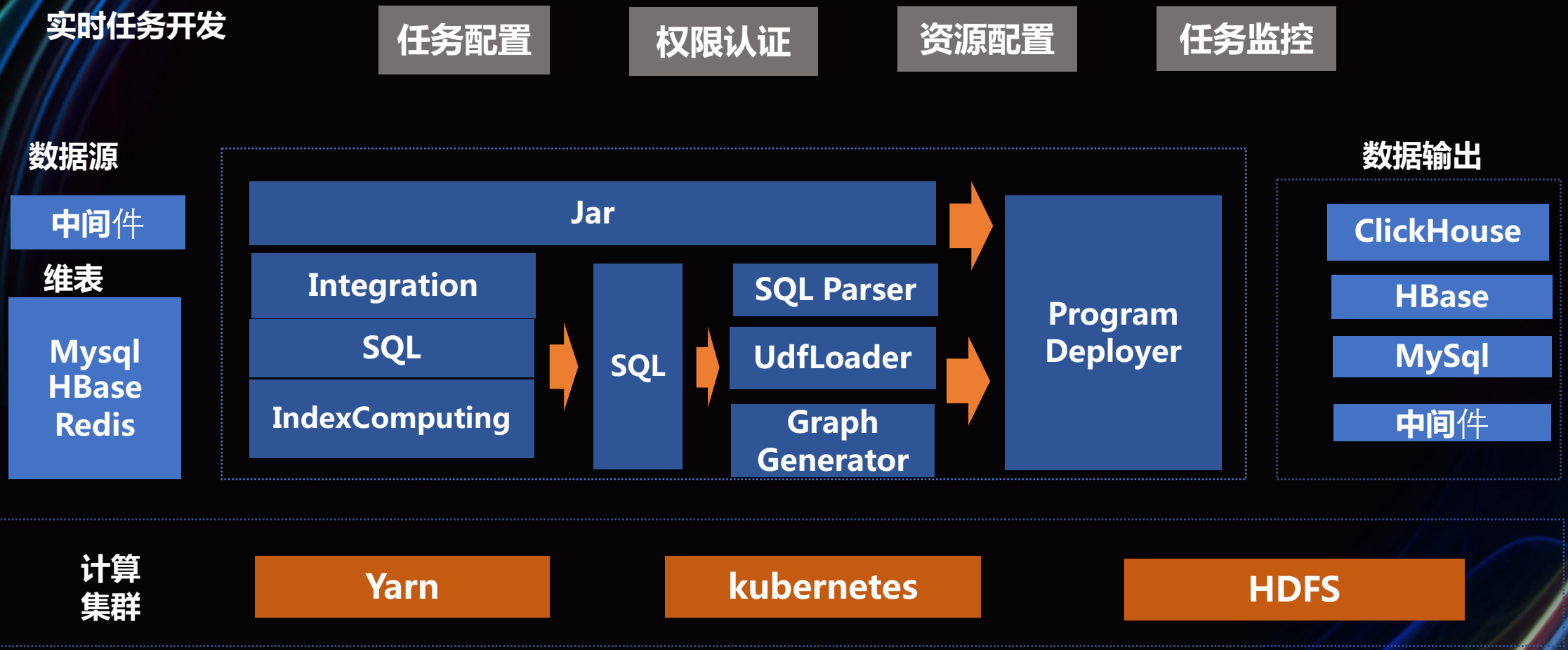
3、Redis集群存储数据安全性问题

4、自定义DSL开发流程，工程开发难度较高

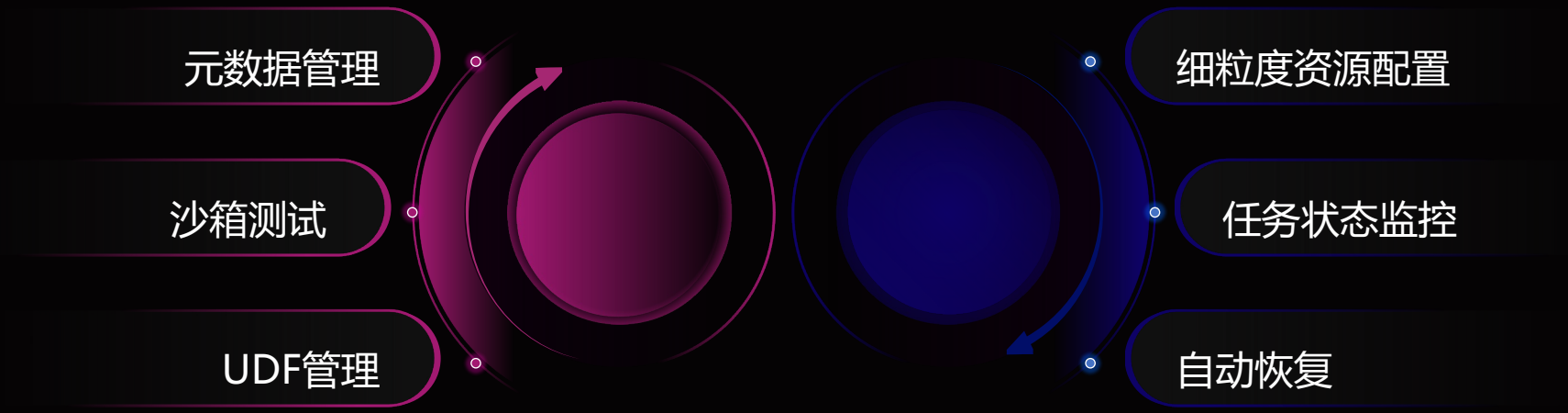
平台架构V2


应用层
实时看板
智能信贷
实时推荐
智慧运营
实时数据开发
IndexComputing 实时指标
SteamingSQL
实时数据集成
存储
ClickHouse
HBase
Mysql
中间件
计算引擎
Flink
批量计算
数据层
业务数据
用户行为数据
集团专区数据
外部数据

开发引擎模块


实时任务开发
任务配置
权限认证
资源配置
任务监控
数据源
中间件
维表
Mysql
HBase
Redis
Jar
Integration
SQL
IndexComputing
SQL
SQL Parser
UdfLoader
Graph Generator
Program Deployer
数据输出
ClickHouse
HBase
MySql
中间件
计算集群
Yarn
kubernetes
HDFS

StreamEngine



一、多SQL任务隔离

目的：

一个任务组多个SQL任务放在单个Job中运行，降低上游kafka的重复消费，提升资源的利用率

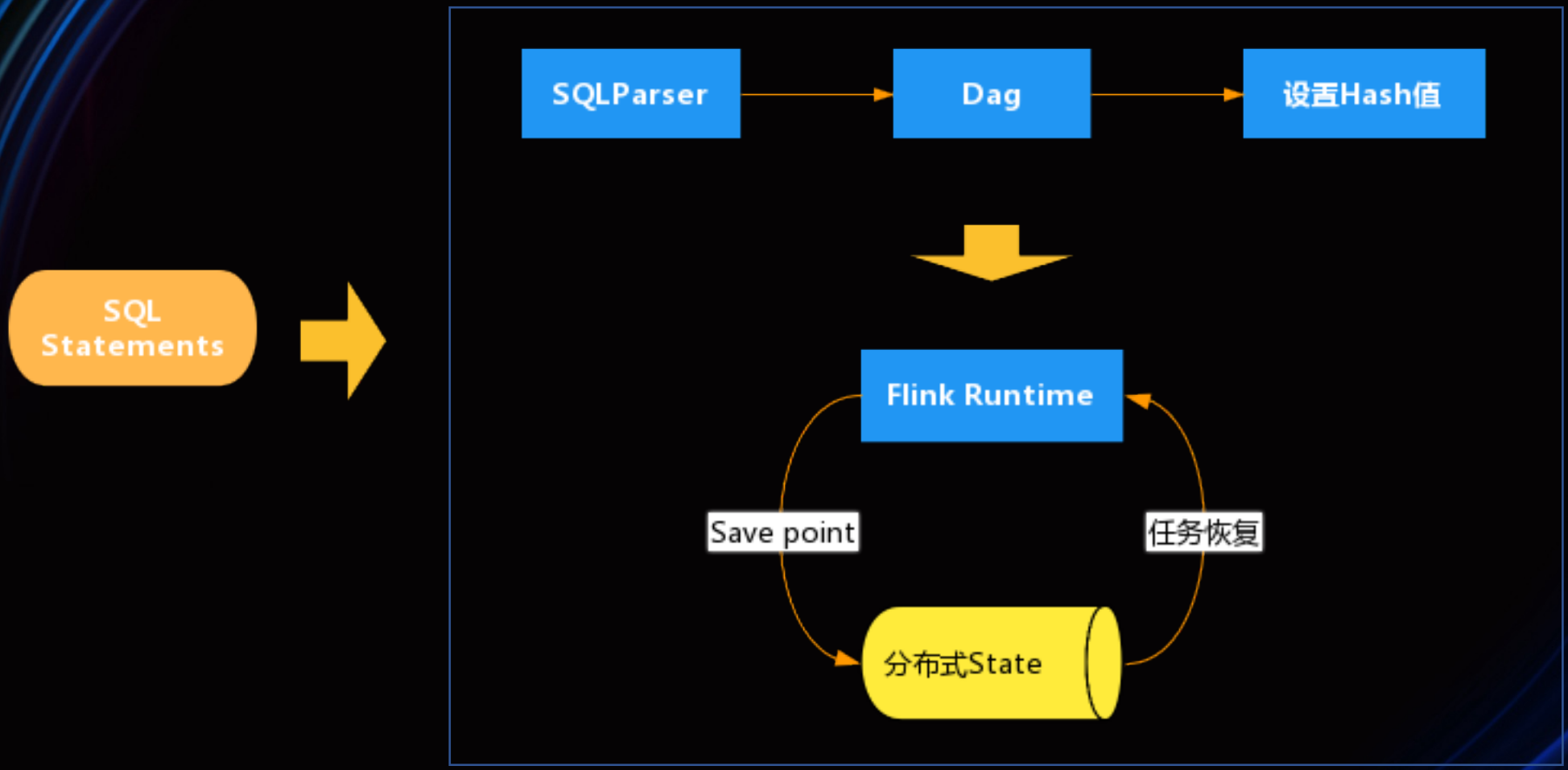
问题：

修改部分SQL其他的FlinkSQL计算任务无法正常从状态恢复

思路：

通过将FlinkSQL的任务的计算拓扑隔离，实现SQL任务的State隔离，从而实现计算任务的恢复隔离

一、多SQL任务隔离


SQL Statements
SQLParser
Dag
设置Hash值
Flink Runtime
Save point
任务恢复
分布式State

一、多SQL任务隔离

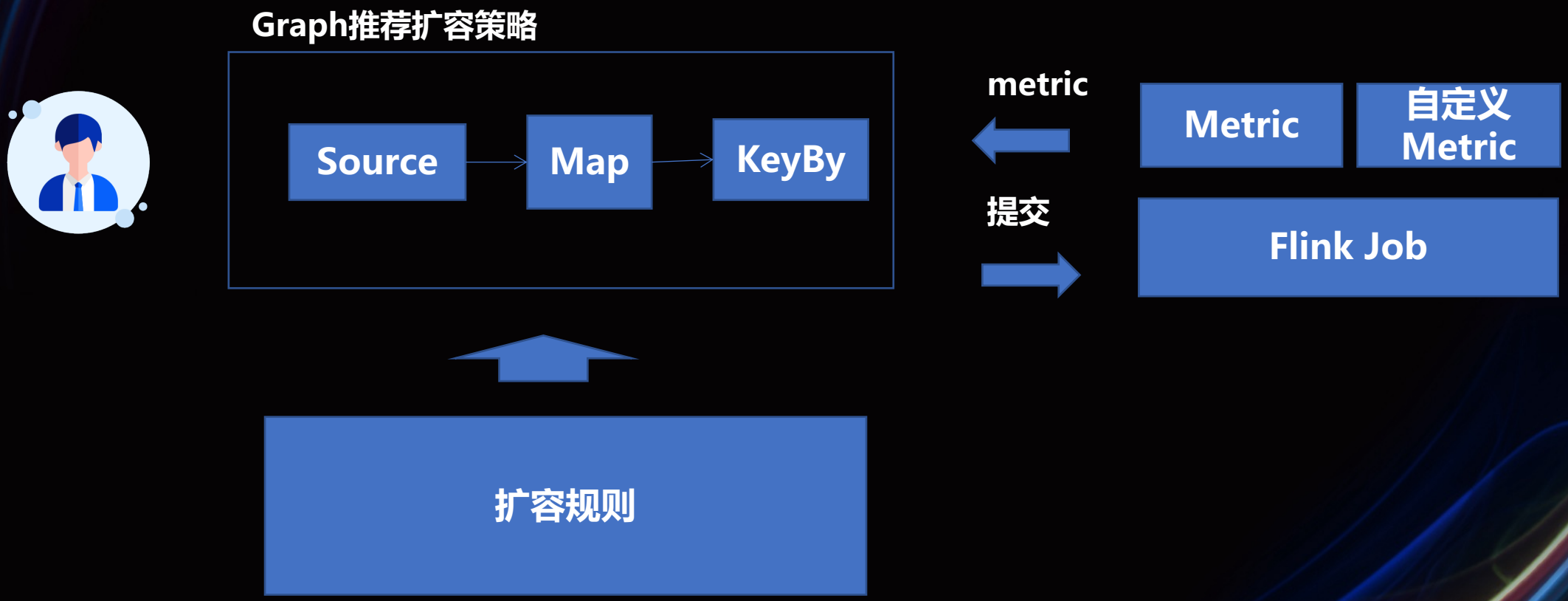
```
try {
    List<String> sqlList = StringUtil.StringSplitToList(sqlsStr);
    Executor executor = lookupExecutor(env);
    Parser parser = ((StreamTableEnvironmentImpl) tableEnv).getParser();
    List<ModifyOperation> modifies = new ArrayList<>();
    List<String> dmlSql = new ArrayList<>();
    tableEnv.getConfig().setSqlDialect(SqlDialect.DEFAULT);
    for (String sql : sqlList) {
        List<Operation> operations = parser.parse(sql);
        Operation op = operations.get(0);
        if (op instanceof ModifyOperation) {
            dmlSql.add(sql);
            modifies.add((ModifyOperation) op);
        } else {
            tableEnv.executeSql(sql);
        }
    }
    List<Transformation<?>> transformationsAll = new ArrayList<>();
    //根据sql递归设置graph hash
    setUpTransformHash(modifies, transformationsAll, tableEnv, dmlSql, sqlIds);
    Pipeline pipeline = executor.createPipeline(transformationsAll, tableEnv.getConfig(), jobName);
    JobClient jobClient = executor.executeAsync(pipeline);
    LOG.info("JobID: {}", jobClient.getJobID());
} catch (Exception e) {
    LOG.info("submit job error: {}", e);
}
```

```
private static void setUpTransformHash(List<ModifyOperation> modifies, List<Transformation<?>> transformationsAll,
                                       StreamTableEnvironment tableEnv, List<String> dmlSqls, String sqlIds) {
    int i = 0;
    List<String> uidList = generateUIDOrArgsGet(dmlSqls,sqlIds);
    for (ModifyOperation operation : modifies) {
        String uid = uidList.get(i);
        List<Transformation<?>> transformations = ((StreamTableEnvironmentImpl) tableEnv).getPlanner()
                .translate(Arrays.asList(operation));
        transformationsAll.addAll(transformations);
        for (Transformation<?> transformation : transformations) {
            transformation.setUid(uid);
            int deep = 0;
            factorial(transformation, uid, deep);
        }
        i++;
    }
}
```

```
private static void factorial(Transformation transformation, String uid, int deep) {
    List<Transformation<?>> transformations = transformation.getInputs();
    uid = uid.concat( str: Constant.UID_DELIMITER + deep);
    if (transformations.size() != 0) {
        for (Transformation tf : transformations) {
            tf.setUid(uid);
            uid = tf.getUid();
            deep += 1;
            factorial(tf, uid, deep);
        }
    }
}
```

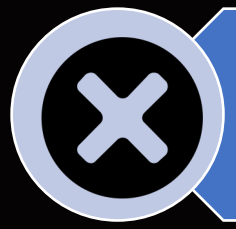
二、SQL任务并行度调整

任务扩容：
针对SQL任务算子扩容，资源不足、进行告警。推荐对Graph进行修改并行度或整体并行度调整、实现并行度扩容。



三、作业调试实践

场景：
用户：希望调试sql运行逻辑，便于开发定位任务异常，以及在生产环境定位数据问题。
开发人员：希望能够得到一个立体的，能够调试数据的运行细节，与中间状态查询。



基于Minicluster实现，替换结果表

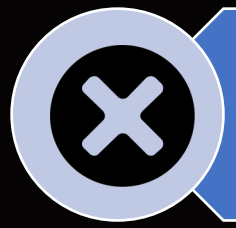
优点：实现流程简单

缺点：不能进行生产环境调试，占用Server资源

基于运行提交YarnCluster，为每个逻辑查询增加中间表

优点：满足开发和生产调试流程

缺点：满足部分生产环境任务调试



镜像任务+底层API ProcessFunction等插装实现

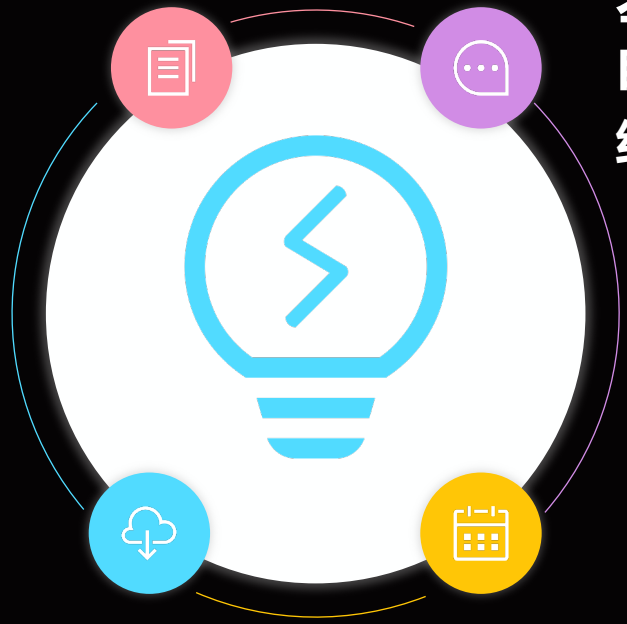
优点：能够立体的观察数据情况，满足开发和生产

缺点：实现难度复杂、消耗一定的计算资源

四、任务监控告警优化

数据的质量监控

提升平台数据质量、及时发现异常



基于规则组进行告警

自定义Metric+Flink Metric组合监控进行任务状态告警

任务分级进行监控告警

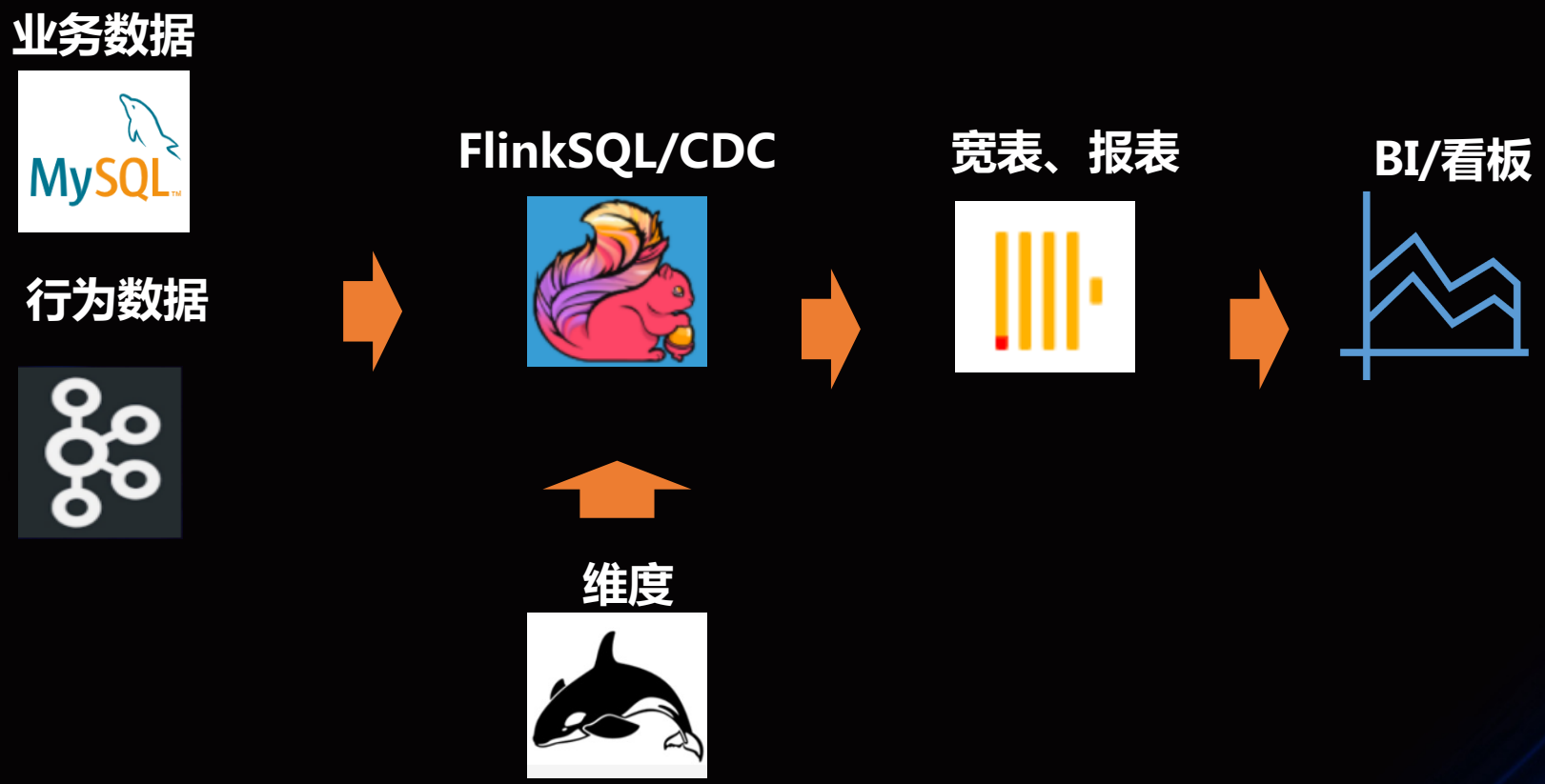
核心任务优先告警、普通任务定期处理

基于规则推荐参数优化

基于任务Metric进行资源优化改进、提前预警资源风险

04 应用场景

一、实时看板



二、湖仓一体探索

场景： 日志、业务数据抽取进入大数据集群

传统架构痛点：

1、日志类数据写入集群小文件

2、业务数据实时性不够

3、表结构经常变更

三、实时集成


TiDB
OCEANBASE
MySQL
ORACLE
FlinkSQL/FlinkCDC
Hudi
HIVE
MySQL
ClickHouse
TiDB

05 未来规划

未来规划

LOREM IPSUM DOLOR

FLINK FORWARD # ASIA 2022 ONLINE | 实时即未来

THANK YOU

谢 谢 观 看