基于Flink和规则引擎的实时风控解决方案

简介： 对一个互联网产品来说，典型的风控场景包括：注册风控、登陆风控、交易风控、活动风控等，而风控的最佳效果是防患于未然，所以事前事中和事后三种实现方案中，又以事前预警和事中控制最好。 这要求风控系统一定要有实时性。

> 案例与解决方案汇总页：
> 阿里云实时计算产品案例&解决方案汇总

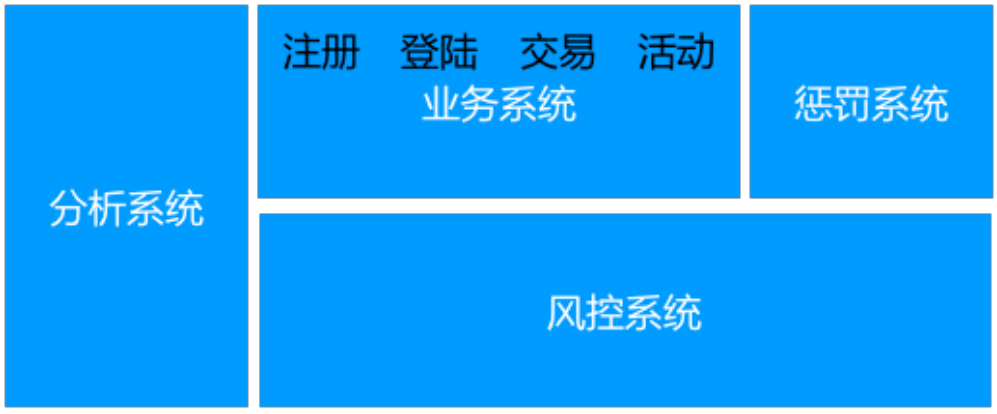
对一个互联网产品来说，典型的风控场景包括：注册风控、登陆风控、交易风控、活动风控等，而风控的最佳效果是防患于未然，所以事前事中和事后三种实现方案中，又以事前预警和事中控制最好。

这要求风控系统一定要有实时性。

本文就介绍一种实时风控解决方案。

1.总体架构

风控是业务场景的产物，风控系统直接服务于业务系统，与之相关的还有惩罚系统和分析系统，各系统关系与角色如下：



- 业务系统，通常是APP+后台 或者 web，是互联网业务的载体，风险从业务系统触发；
- 风控系统，为业务系统提供支持，根据业务系统传来的数据或埋点信息来判断当前用户或事件有无风险；
- 惩罚系统，业务系统根据风控系统的结果来调用，对有风险的用户或事件进行控制或惩罚，比如增加验证码、限制登陆、禁止下单等等；
- 分析系统，该系统用以支持风控系统，根据数据来衡量风控系统的表现，比如某策略拦截率突然降低，那可能意味着该策略已经失效，又比如活动商品被强光的时间突然变短，这表面总体活动策略可能有问题等

等，该系统也应支持运营/分析人员发现新策略；

其中**风控系统**和**分析系统**是本文讨论的重点，而为了方便讨论，我们假设业务场景如下：

- 电商业务；
- 风控范围包括：
  - 注册，虚假注册；
  - 登陆，盗号登陆；
  - 交易，盗刷客户余额；
  - 活动，优惠活动薅羊毛；
- 风控实现方案：事中风控，目标为拦截异常事件；

2.风控系统

风控系统有**规则**和**模型**两种技术路线，规则的优点是简单直观、可解释性强、灵活，所以长期活跃在风控系统之中，但缺点是容易被攻破，一但被黑产猜到里面就会失效，于是在实际的风控系统中，往往再结合上基于模型的风控环节来增加健壮性。但限于篇幅，本文中我们只重点讨论一种基于规则的风控系统架构，当然如果有模型风控的诉求，该架构也完全支持。

规则就是针对事物的条件判断，我们针对注册、登陆、交易、活动分别假设几条规则，比如：

- 用户名与身份证姓名不一致；
- 某IP最近1小时注册账号数超过10个；
- 某账号最近3分钟登陆次数大于5次；
- 某账号群体最近1消失购买优惠商品超过100件；
- 某账号最近3分钟领券超过3张；

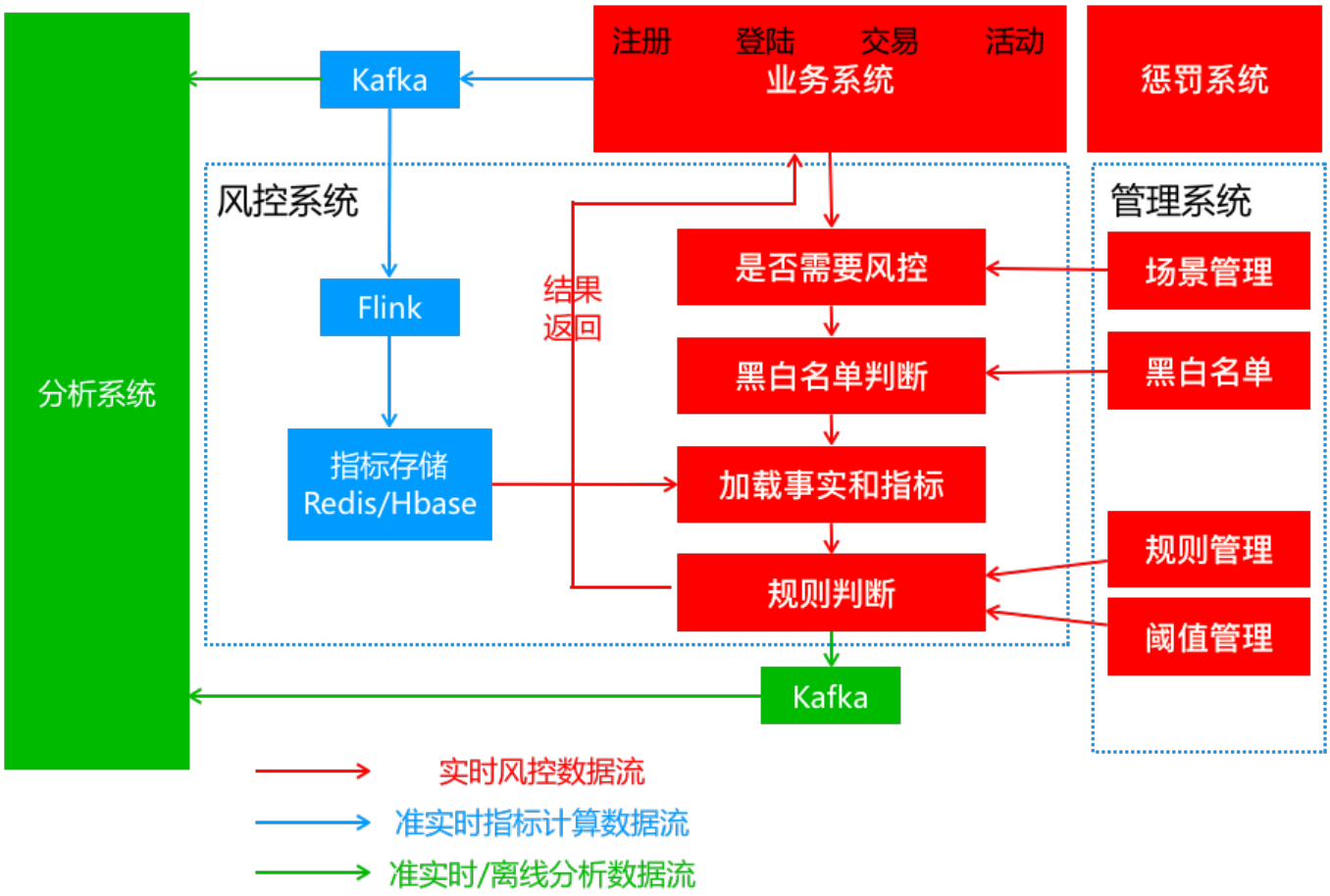
规则可以组合成规则组，为了简单起见，我们这里只讨论规则。

规则其实包括三个部分：

- 事实，即被判断的主体和属性，如上面规则的账号及登陆次数、IP和注册次数等；
- 条件，判断的逻辑，如某事实的某属性大于某个指标；
- 指标阈值，判断的依据，比如登陆次数的临界阈值，注册账号数的临界阈值等；

规则可由运营专家凭经验填写，也可由数据分析师根据历史数据发掘，但因为规则在与黑产的攻防之中会被猜中导致失效，所以无一例外都需要动态调整。

基于上边的讨论，我们设计一个风控系统方案如下：



该系统有三条数据流向：

- 实时风控数据流，由红线标识，同步调用，为风控调用的核心链路；
- 准实时指标数据流，由蓝线标识，异步写入，为实时风控部分准备指标数据；
- 准实时/离线分析数据流，由绿线标识，异步写入，为风控系统的表现分析提供数据；

本节先介绍前两部分，分析系统在下一节介绍。

2.1 实时风控

实时风控是整个系统的核心，被业务系统同步调用，完成对应的风控判断。

前面提到规则往往由人编写并且需要动态调整，所以我们会把风控判断部分与规则管理部分拆开。规则管理后台为运营服务，由运营人员去进行相关操作：

- 场景管理，决定某个场景是否实施风控，比如活动场景，在活动结束后可以关闭该场景；
- 黑白名单，人工/程序找到系统的黑白名单，直接过滤；
- 规则管理，管理规则，包括增删或修改，比如登陆新增IP地址判断，比如下单新增频率校验等；
- 阈值管理，管理指标的阈值，比如规则为某IP最近1小时注册账号数不能超过10个，那1和10都属于阈值；

讲完管理后台，那规则判断部分的逻辑也就十分清晰了，分别包括前置过滤、事实数据准备、规则判断三个环节。

2.1.1 前置过滤

业务系统在特定事件（如注册、登陆、下单、参加活动等）被触发后同步调用风控系统，附带相关上下文，比如IP地址，事件标识等，规则判断部分会根据管理后台的配置决定是否进行判断，如果是，接着进行黑白名单过滤，都通过后进入下一个环节。

这部分逻辑非常简单。

2.1.2 实时数据准备

在进行判断之前，系统必须要准备一些事实数据，比如：

- 注册场景，假如规则为单一IP最近1小时注册账号数不超过10个，那系统需要根据IP地址去redis/hbase找到该IP最近1小时注册账号的数目，比如15；
- 登陆场景，假如规则为单一账号最近3分钟登陆次数不超过5次，那系统需要根据账号去redis/hbase找到该账号最近3分钟登陆的次数，比如8；

redis/hbase的数据产出我们会在第2.2节准实时数据流中进行介绍。

2.2.3 规则判断

在得到事实数据之后，系统会根据规则和阈值进行判断，然后返回结果，整个过程便结束了。

整个过程逻辑上是清晰的，我们常说的规则引擎主要在这部分起作用，一般来说这个过程有两种实现方式：

- 借助成熟的规则引擎，比如Drools，Drools和Java环境结合的非常好，本身也非常完善，支持很多特性，不过使用比较繁琐，有较高门槛，可参考文章【1】；
- 基于Groovy等动态语言自己完成，这里不做赘述。可参考文章【2】；

这两种方案都支持规则的动态更新。

2.2 准实时数据流

这部分属于后台逻辑，为风控系统服务，准备事实数据。

把数据准备与逻辑判断拆分，是出于系统的性能/可扩展性的角度考虑的。

前边提到，做规则判断需要事实的相关指标，比如最近一小时登陆次数，最近一小时注册账号数等等，这些指标通常有一段时间跨度，是某种状态或聚合，很难在实时风控过程中根据原始数据进行计算，因为风控的规则引擎往往是无状态的，不会记录前面的结果。

同时，这部分原始数据量很大，因为用户活动的原始数据都要传过来进行计算，所以这部分往往由一个流式大数据系统来完成。在这里我们选择Flink，Flink是当今流计算领域无可争议的No.1，不管是性能还是功能，都能很好的完成这部分工作。

这部分数据流非常简单：

- 业务系统把埋点数据发送到Kafka；

- Flink订阅Kafka，**完成原子粒度的聚合**；
    - > 注：Flink仅完成原子粒度的聚合是和规则的动态变更逻辑相关的。举例来说，在注册场景中，运营同学会根据效果一会要判断某IP最近1小时的注册账号数，一会要判断最近3小时的注册账号数，一会又要判断最近5小时的注册账号数……也就是说这个最近N小时的N是动态调整的。那Flink在计算时只应该计算1小时的账号数，在判断过程中根据规则来读取最近3个1小时还是5个1小时，然后聚合后进行判断。因为在Flink的运行机制中，作业提交后会持续运行，如果调整逻辑需要停止作业，修改代码，然后重启，相当麻烦；同时因为Flink中间状态的问题，重启还面临着中间状态能否复用的问题。所以假如直接由Flink完成N小时的聚合的话，每次N的变动都需要重复上面的操作，有时还需要追数据，非常繁琐。
- Flink把汇总的指标结果写入Redis或Hbase，供实时风控系统查询。两者问题都不大，根据场景选择即可。

通过把数据计算和逻辑判断拆分开来并引入Flink，我们的风控系统可以应对极大的用户规模。

3.分析系统

前面的东西静态来看是一个完整的风控系统，但动态来看就有缺失了，这种缺失不体现在功能性上，而是体现在演进上。即如果从动态的角度来看一个风控系统的话，我们至少还需要两部分，一是衡量系统的整体效果，一是为系统提供规则/逻辑升级的依据。

在衡量整体效果方面，我们需要：

- 判断规则是否失效，比如拦截率的突然降低；
- 判断规则是否多余，比如某规则从来没拦截过任何事件；
- 判断规则是否有漏洞，比如在举办某个促销活动或发放代金券后，福利被领完了，但没有达到预期效果；

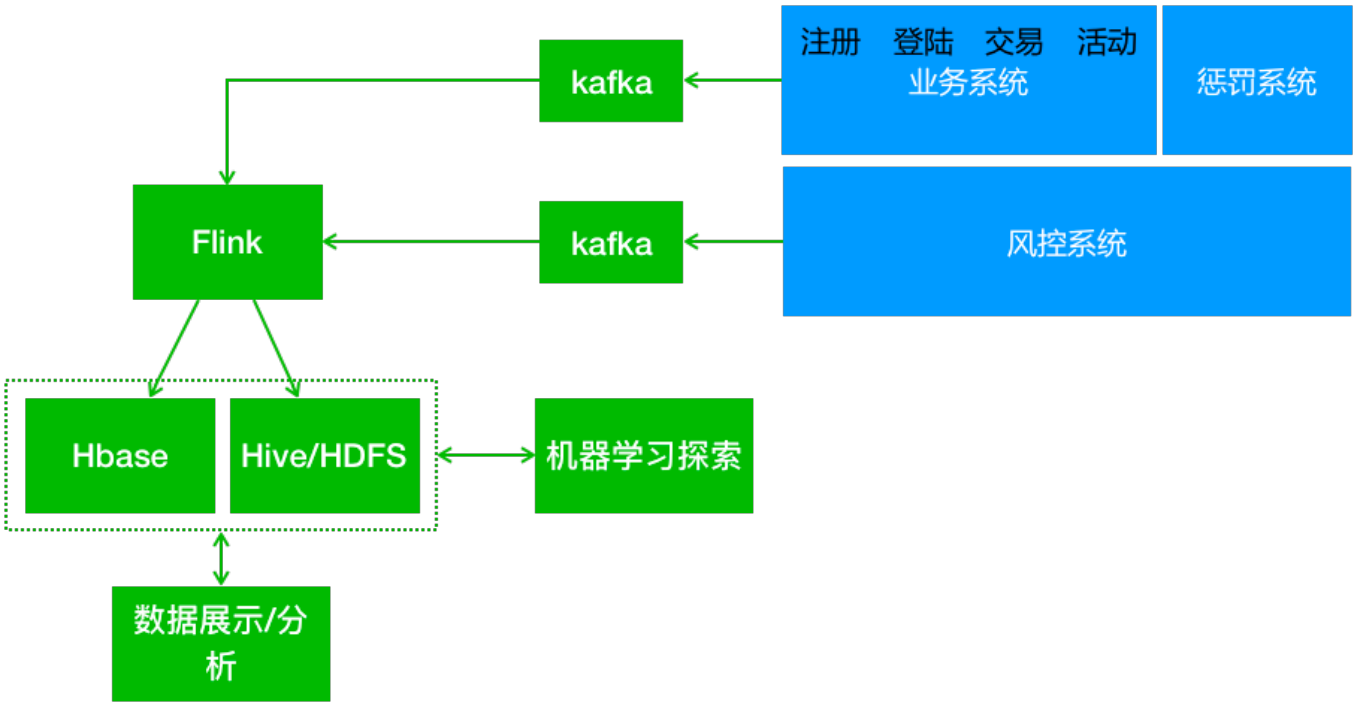
在为系统提供规则/逻辑升级依据方面，我们需要：

- 发现全局规则，比如某人在电子产品的花费突然增长了100倍，单独来看是有问题的，但整体来看，可能很多人都出现了这个现象，原来是苹果发新品了……
- 识别某种行为的组合，单次行为是正常的，但组合是异常的，比如用户买菜刀是正常的，买车票是正常的，买绳子也是正常的，去加油站加油也是正常的，但短时间内同时做这些事情就不是正常的。
- 群体识别，比如通过图分析技术，发现某个群体，然后给给这个群体的所有账号都打上群体标签，防止出现那种每个账号表现都正常，但整个群体却在集中薅羊毛的情况。

这便是分析系统的角色定位，在他的工作中有部分是确定性的，也有部分是探索性的，为了完成这种工作，该系统需要尽可能多的数据支持，如：

- 业务系统的数据，业务的埋点数据，记录详细的用户、交易或活动数据；
- 风控拦截数据，风控系统的埋点数据，比如某个用户在具有某些特征的状态下因为某条规则而被拦截，这条拦截本身就是一个事件数据；

这是一个典型的大数据分析场景，架构也比较灵活，我仅仅给出一种建议的方式。



相对来说这个系统是最开放的，既有固定的指标分析，也可以使用机器学习/数据分析技术发现更多新的规则或模式，限于篇幅，这里就不详细展开了。

4.参考资料

1.从 Drools 规则引擎到风控反洗钱
2.基于Groovy的规则脚本引擎实战
3.基于规则的风控系统
4.网易严选风控实践
5.网易考拉规则引擎平台架构设计与实践
6.一个开源java风控系统