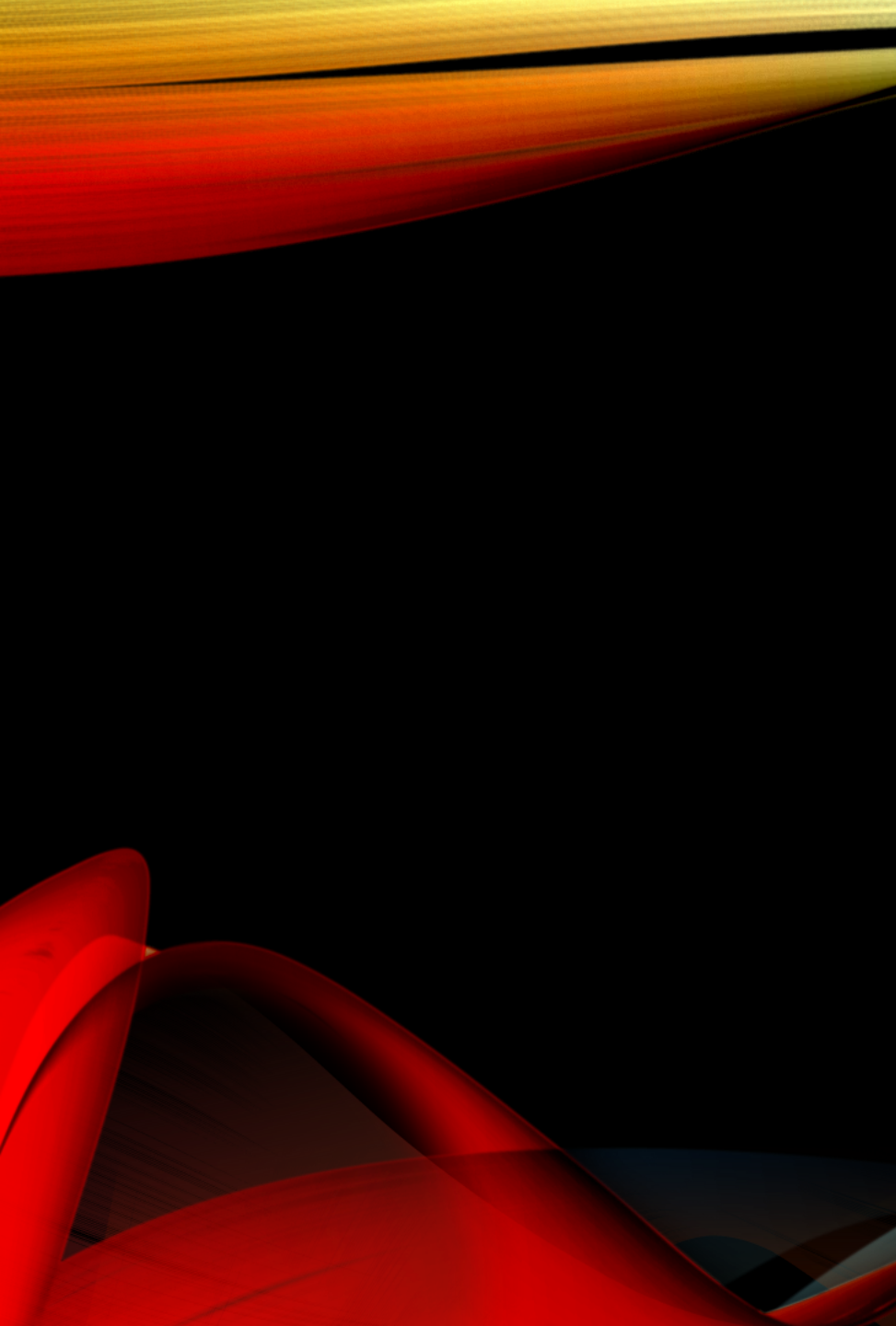


实时数据流处理的难点

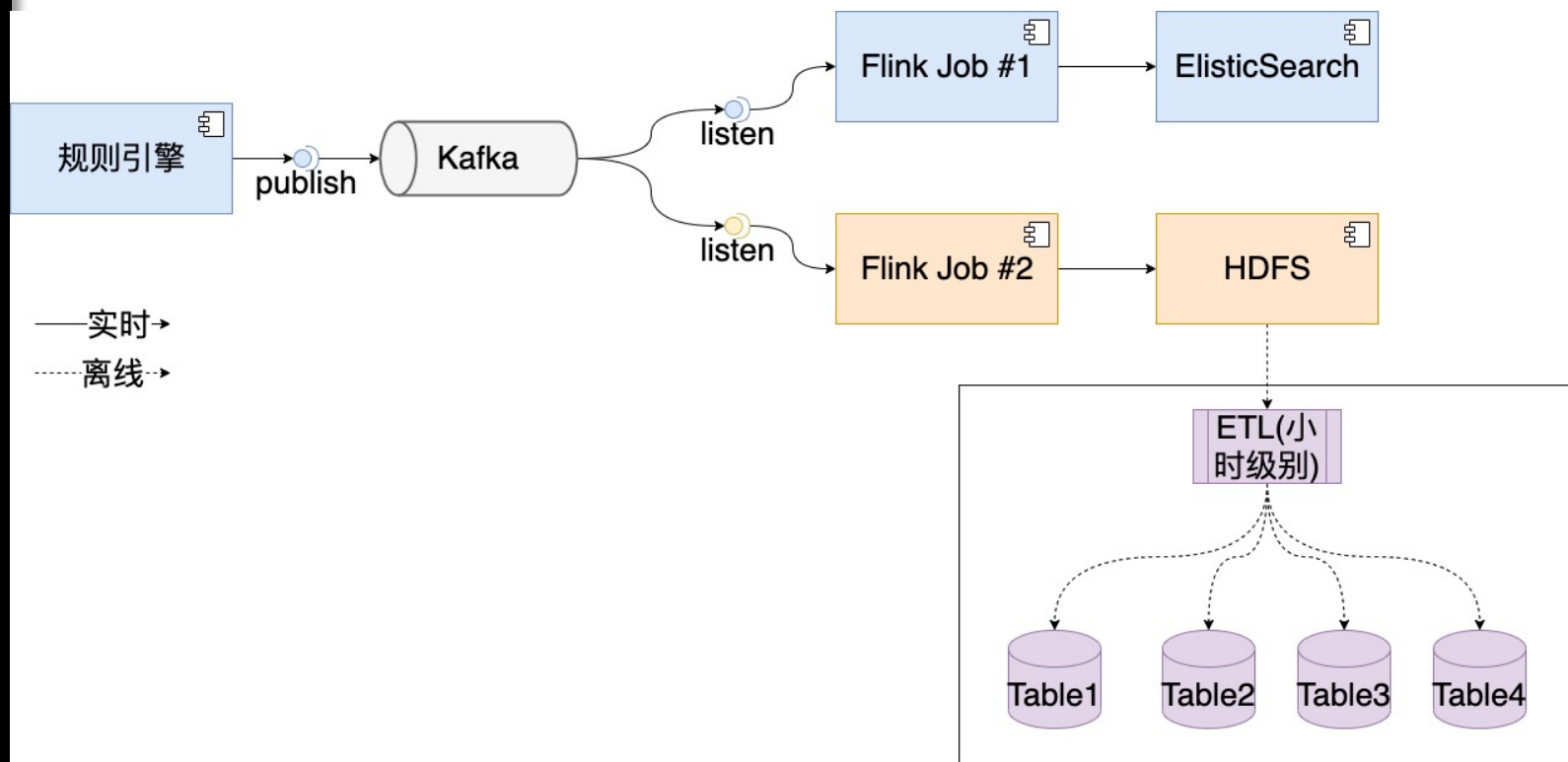


公司的规则引擎每天都会产生数以亿计的核心业务日志，下游分析师会严重依赖该日志数据分析、统计每个规则的质量、执行效率等信息。由于该业务日志数据量以及单条数据size都很大，所以无法直接写入RDBMS，而只能写入Kafka，然后由消费者进行消费。目前有两个消费者写入两个存储，7天内的数据写入ElasticSearch，永久数据写入HDFS。考虑到时效性，HDFS中的数据至少要在1小时内刷新。

topic有50个producer，每天snappy压缩后的数据量是7T~10T，单条Avro数据size在128KB以上。

技术栈：Flink、Kafka、HDFS、SparkSQL

- 中规中矩的架构方案
- 难点在哪里？坑在哪里？



史上最难

- Avro数据解析
 - 1万条数据，解析需要10秒以上
 - SparkSQL处理每天数据需要2万+个MAP，运行时间超过3小时
 - 结构复杂，包含Avro的所有数据类型及其嵌套，特别是含有Union数据类型
 - producer升级时，会同时存在2个Schema，由于某种原因，Schema以文件格式手动下发给消费者
 - 解析性能瓶颈，在于Java的new String(40%~80%)

- Kafka限制
 - 数据量太大，Topic对应的保留时间只有3天，解析速度跟不上，容易导致消费不及时丢数据
 - producer写入Kafka之后，没有其他数据源可供查询，丢数据时，无法通过Kafka补数据

- Flink Job

- 平台资源不够，无法申请到足够的JobManager/TaskManager，及其CPU/MEMORY
- StreamingFileSink写入的文件名格式为part-
<subtaskIndex>-<partFileIndex>，无法添加唯一的标志。极端情况下（savepoint/checkpoint丢失等），Flink Job会完全重新启动，导致文件名从part-0-0开始，容易覆盖原文件

- HDFS/SparkSQL
 - 由于平台限制，目标表必须是Parquet格式，Avro无法直接实时落地供用户查询
 - SparkSQL无法读取union数据（版本问题）
 - 如何去重

- 拆表

- 原数据结构复杂，使用起来比较困难，需要拆分成4个部分，存到4个表
- Avro数据解析慢，无法直接在Flink中进行拆分，只能用SparkSQL
- 每个表需要按照天&小时分区
- 读取Avro格式的表时，能否只解析部分字段（例如前8个字段），以便加快速度

- 离线、实时衔接
 - 离线读取实时写入的文件，不能包含重复数据
 - 实时写入的文件不能覆盖离线的文件

- 丢数据
 - 如何确定丢数据
 - 如何确定丢失哪些数据
 - 数据丢失后，如何及时、精确（不重复）的补数据

难点8

- 监控
 - 端到端监控如何做: Kafka和target表
 - Kafka消息的Lag如何度量

难点

- 简单来说，都是数据量太大、单条数据size太大惹的祸。如果这个问题不存在，则后面所有的难点基本不存在

Q & A

- 敢于问一个好问题，大概率能解决问题的90%

