

日志分析变迁史

-- 51信用卡

管你多少信用卡

电话银行，查余额，查账单，一键直达
银行账单，无需导入，已经支持36家银行账单自动解析

目录

- ↓ DB的select count (2012)
- ↓ 实时日志分析 (2013)
 - 同步方式 (Redis)
 - 异步方式 (MongoDB)
- ↓ 大数据分析 (2014)
 - 行为日志 (HDFS/Hive/Map-Reduce)
 - 实时分析(Storm)
- ↓ 数据分析平台化 (2015)
- ↓ 引入规则引擎 (2015)

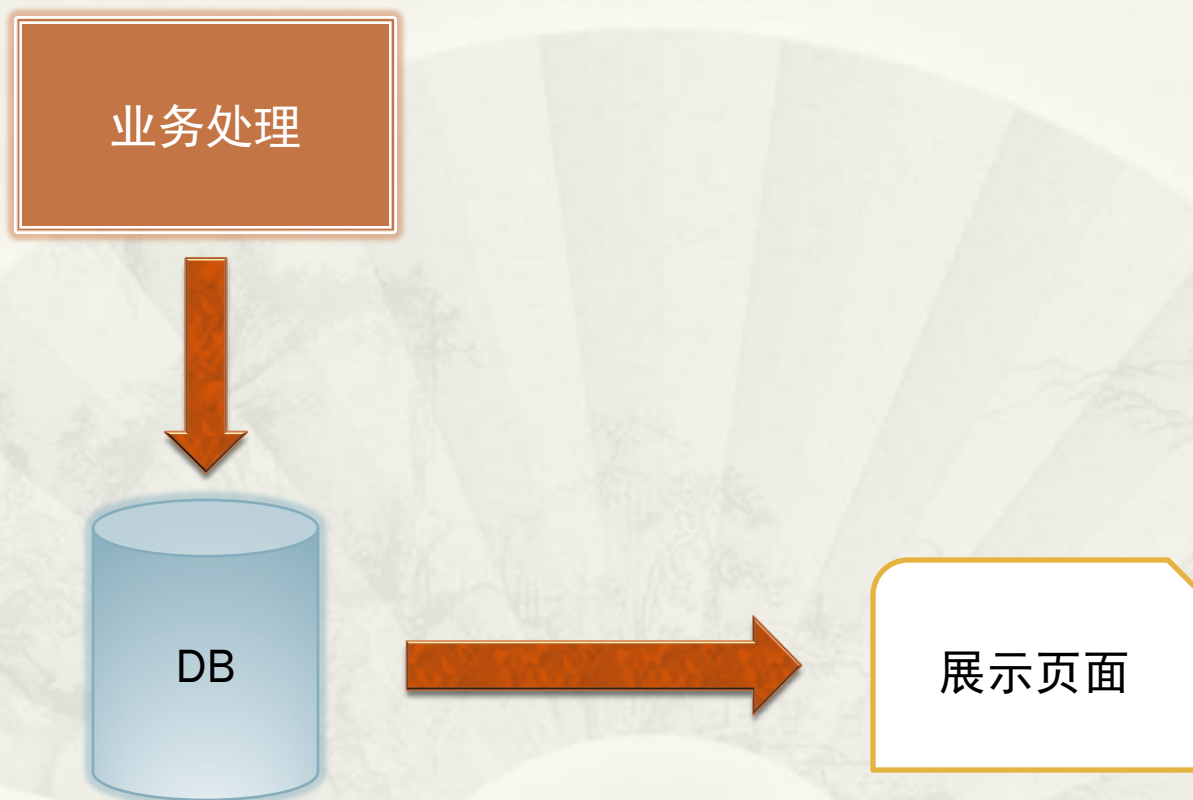


DB的SELECT COUNT

DB的select count

- ⇓ 原始数据积累
- ⇓ 仅仅关注每天新增用户、新增邮箱等各种总数
- ⇓ 50万用户内，5台服务器；
- ⇓ 整个系统全是同步处理

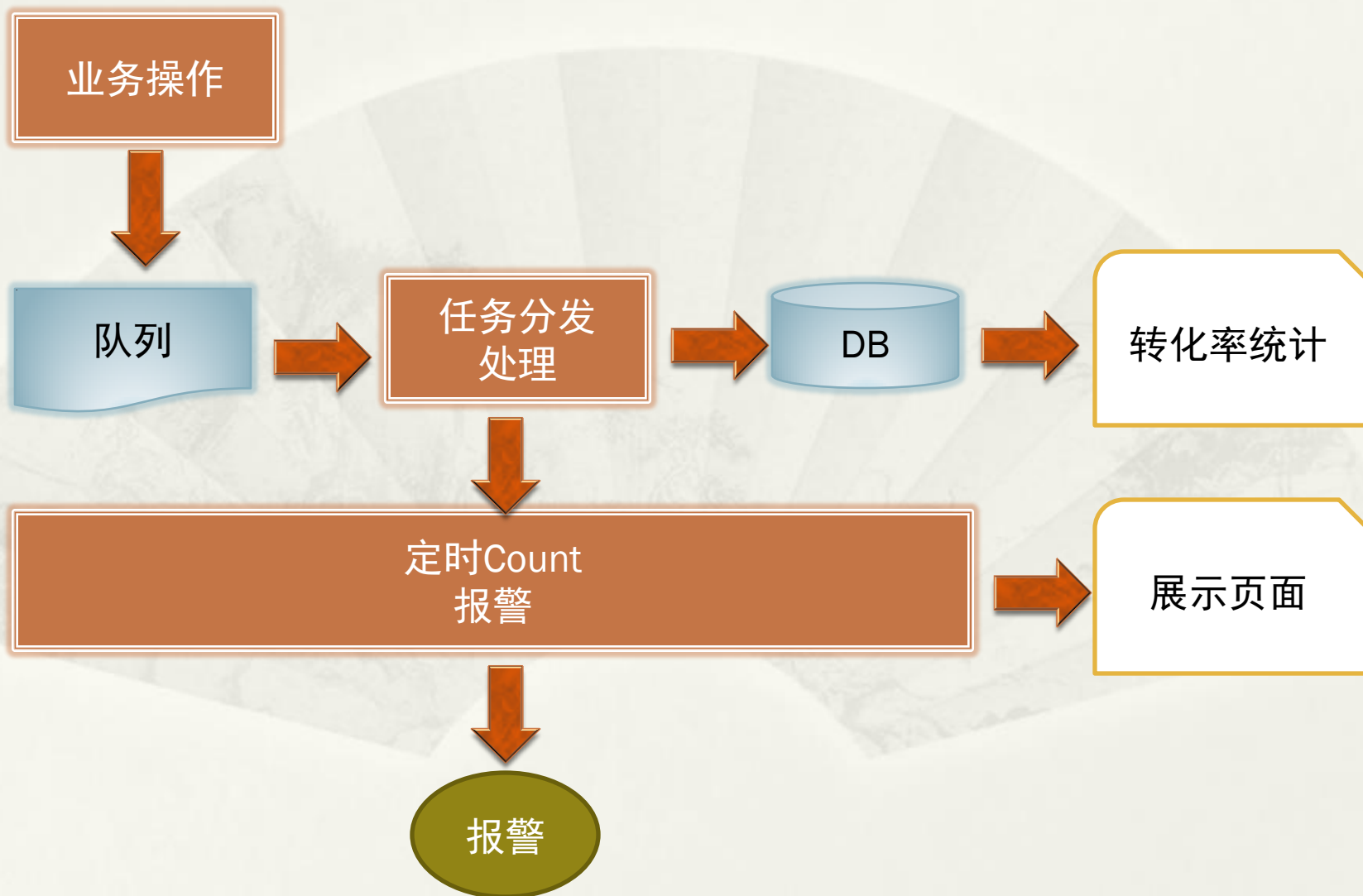
DB的select count



DB的select count (异步)

- ⇓ 200万用户，20台服务器
- ⇓ 整个系统改造成异步处理
- ⇓ 主要关注指标是应用层故障指标、同步报警、各新增指标
- ⇓ 开始关注转化率 (DB count, 定时运行)

DB的select count (异步)



日志分析从进程内同步队列到异步方式两阶段

日志分析

日志数据分析背景案例

↓ 今天邮箱转化率低了？有效用户成本增加。

↓ 1.查数据走势

- ✓ 1.1.新增设备总数增加
- ✓ 1.2.注册用户总数增加
- ✓ 1.3.导入用户总数增加

结论数据良好，但成本增加

↓ 2.查问题

- 2.1.查转化率，发现新增设备到导入用户各转化率下降厉害；
- 2.2.查各渠道转化率，发现积分墙推广的渠道问题。

基于第三方平台仅仅后发现、后分析，需要实时报警、精准推算

实时日志分析背景

基于Count的缺点

- 实时数据库Count无中间数据，无法同比分析
- 和业务服务耦合，系统性能消耗大
- 交叉分析困难
- 各业务线统计太过分散，重复造轮子，口径不一致算法各异

目标

- 分离统计和业务服务
- 实时预警（系统稳定、安全攻击）
- 运营、产品数据及其转化率
- 统一数据分析口径
- 积累基础数据、中间数据

实时分析内容

⇓ 运营、产品层面

- 用户数（如：注册用户瞬间变少）
- 访问数
- 转化率
- 渠道

⇓ 安全、技术层面

- 服务器、邮件服务
- 业务服务
- 网络
- 访问请求

实时数据监控

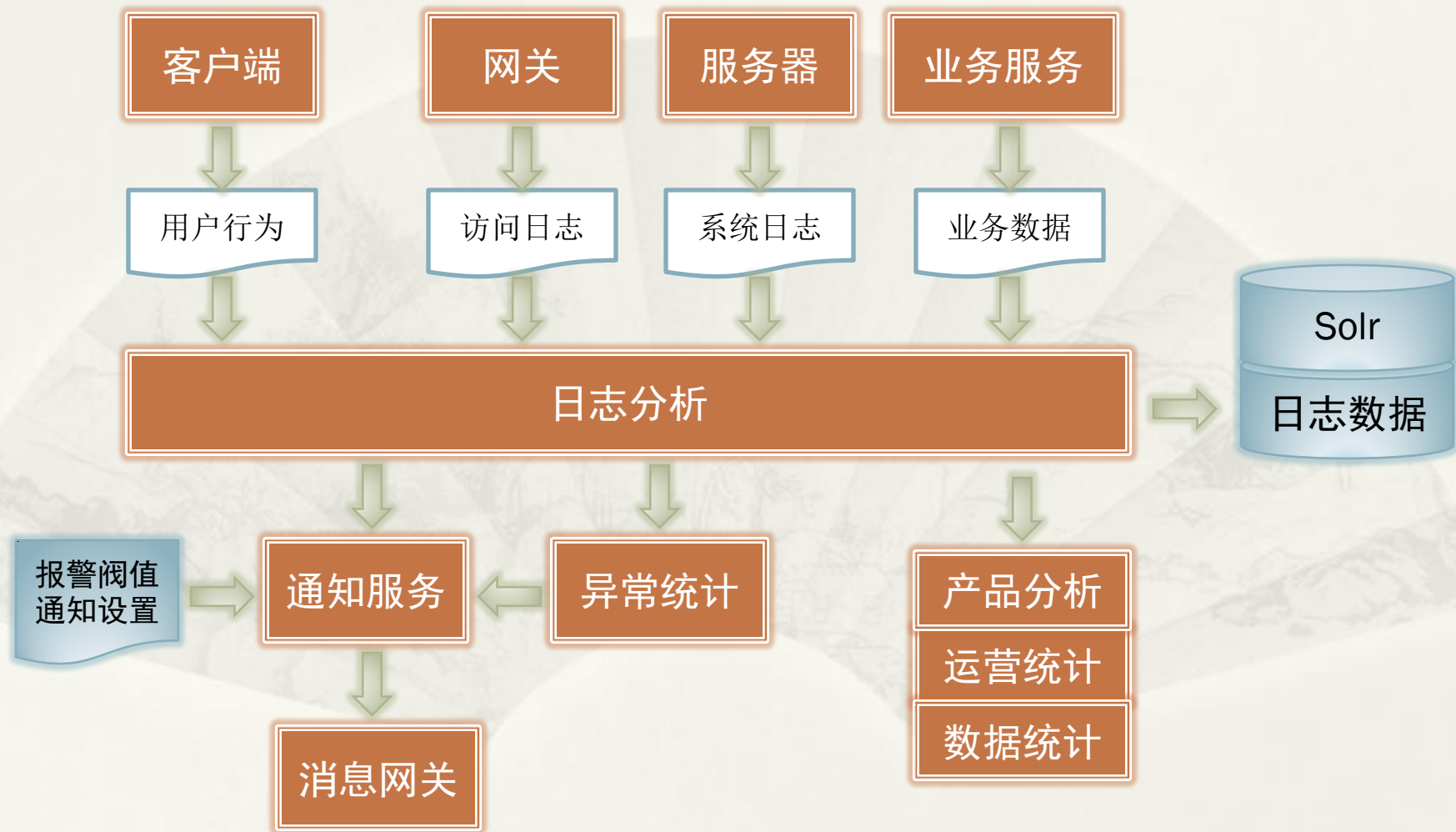
业务数据监控

- ↓ 按小时采用最近7天平均，加上浮动，作为阈值
- ↓ 监控值：总数、百分比变化
- ↓ 监控纬度：地域、新增、处理数、转化率

系统数据监控

- ↓ http:5xx、4xx
- ↓ 网络:tps、流量
- ↓ 用户:单用户请求数、单设备请求数
- ↓ 服务:rt、可用性、IO

日志分析逻辑设计





JVM内存的队列

同步日志分析

同步日志分析逻辑设计

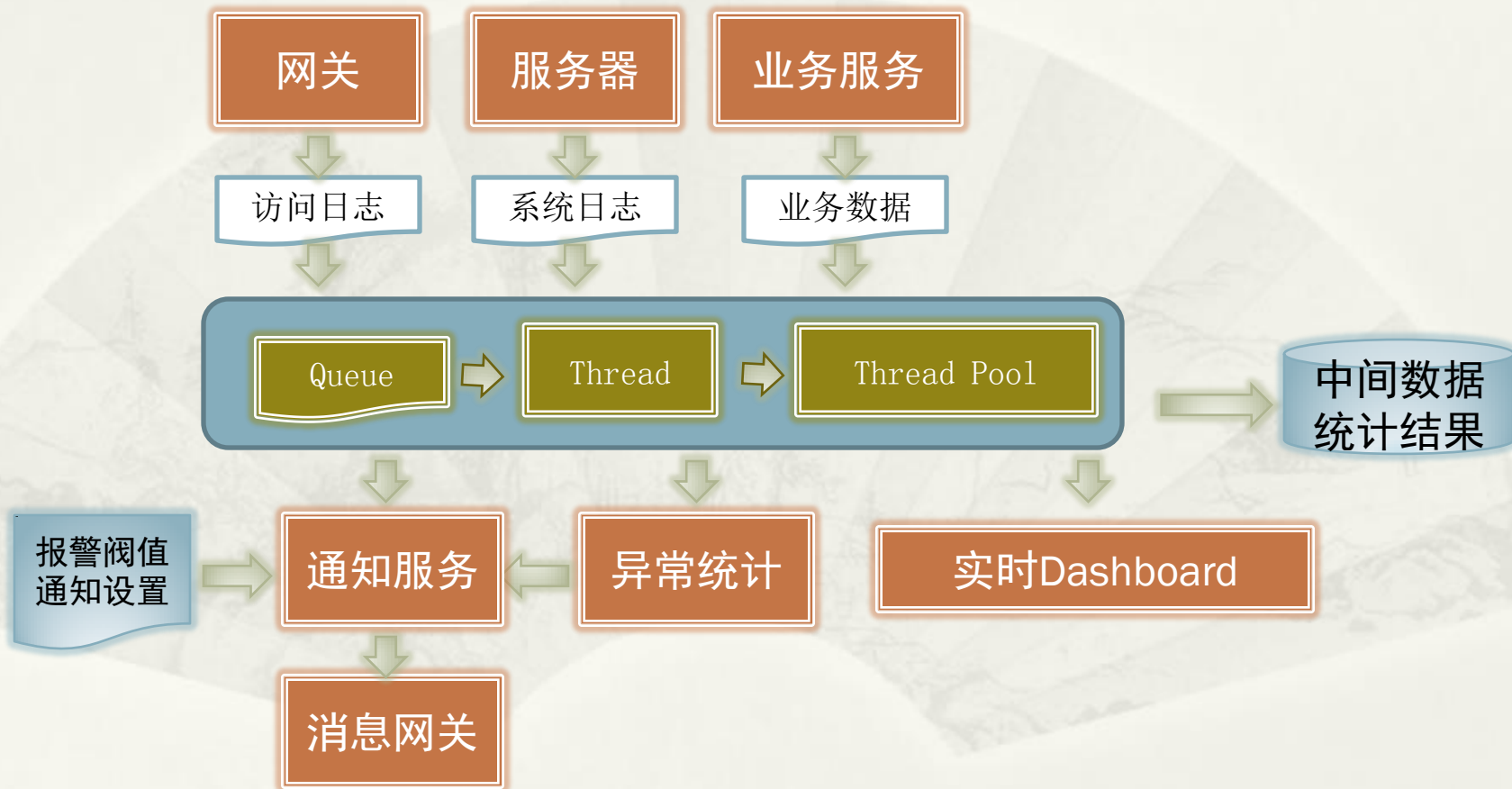
⇓ 设计思路

- 完全使用JVM内存保存瞬时数据
- 使用线程池保证异步处理

⇓ 存在问题

- 并发峰值加大，很容易内存溢出
- 线程池过大，容易造成线程死锁
- 一旦异常崩溃，丢失数据严重，且无法恢复
- 数据高峰和低谷期的资源相差太大，造成严重资源浪费
- 所有数据统一级别，无法做降级处理

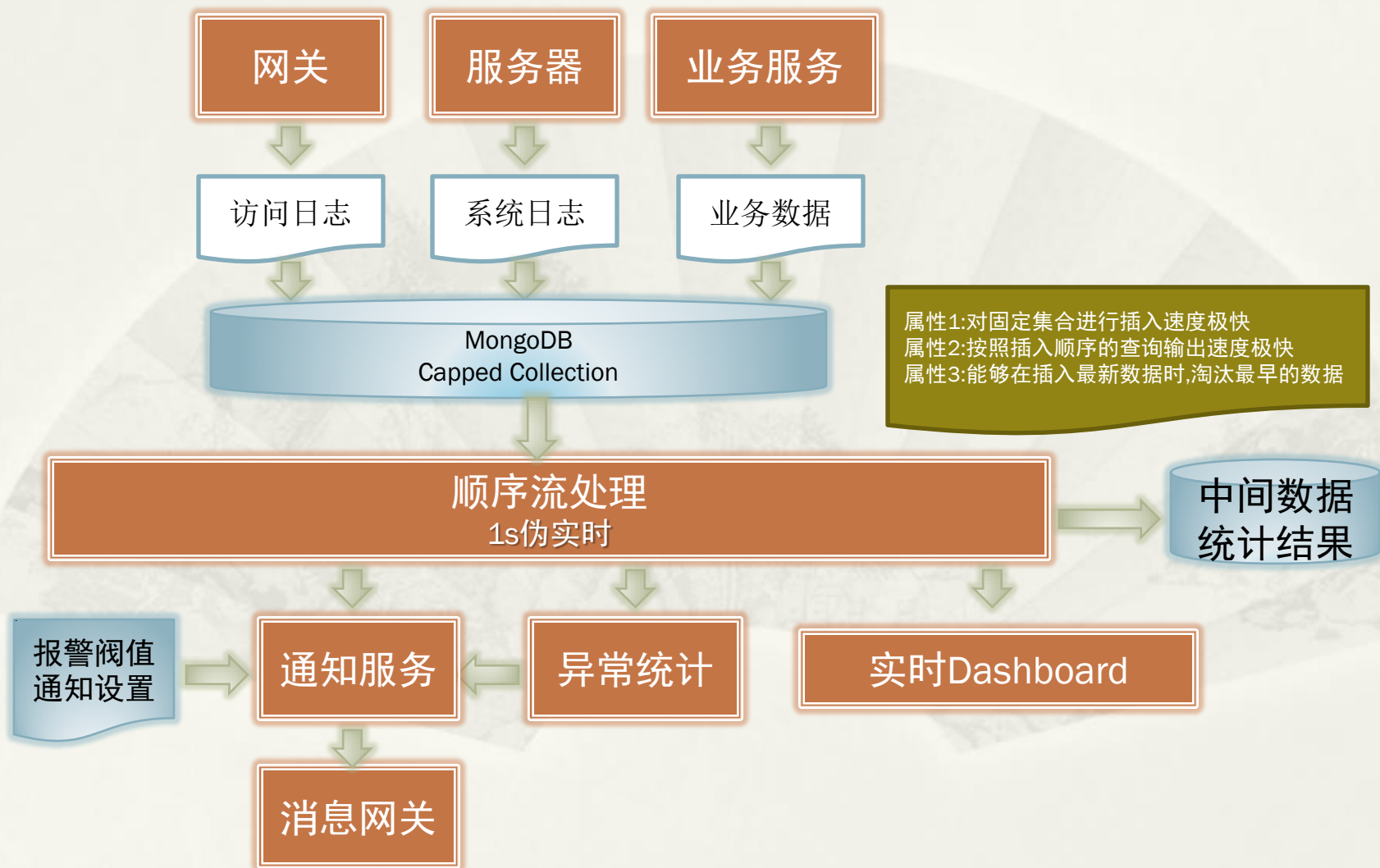
同步日志分析逻辑设计



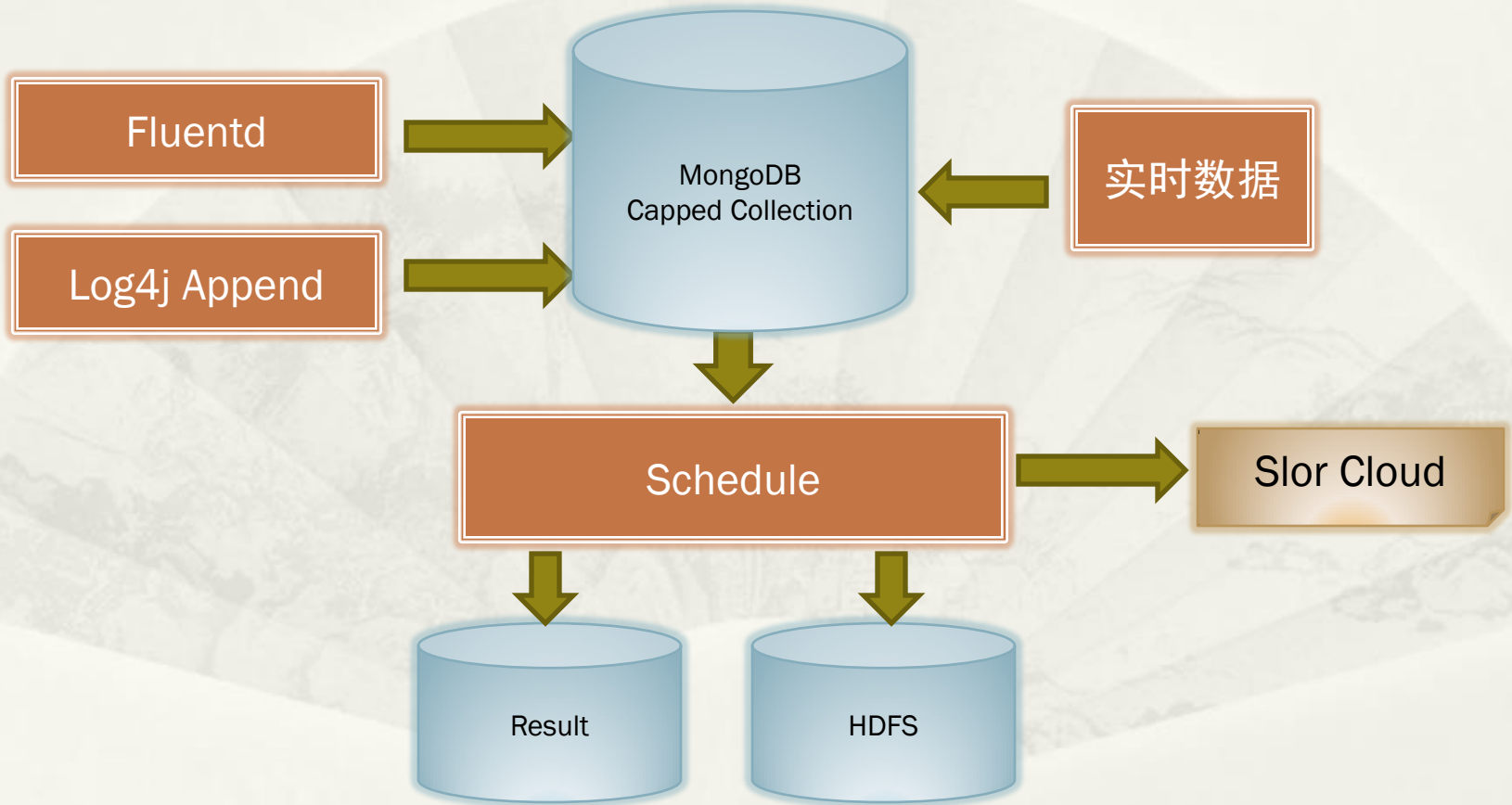
MongoDB

异步日志分析

异步数据分析逻辑设计



日志收集



大数据 (hadoop)

行为日志分析

行为日志分析背景

⇓ 基于第三方统计平台

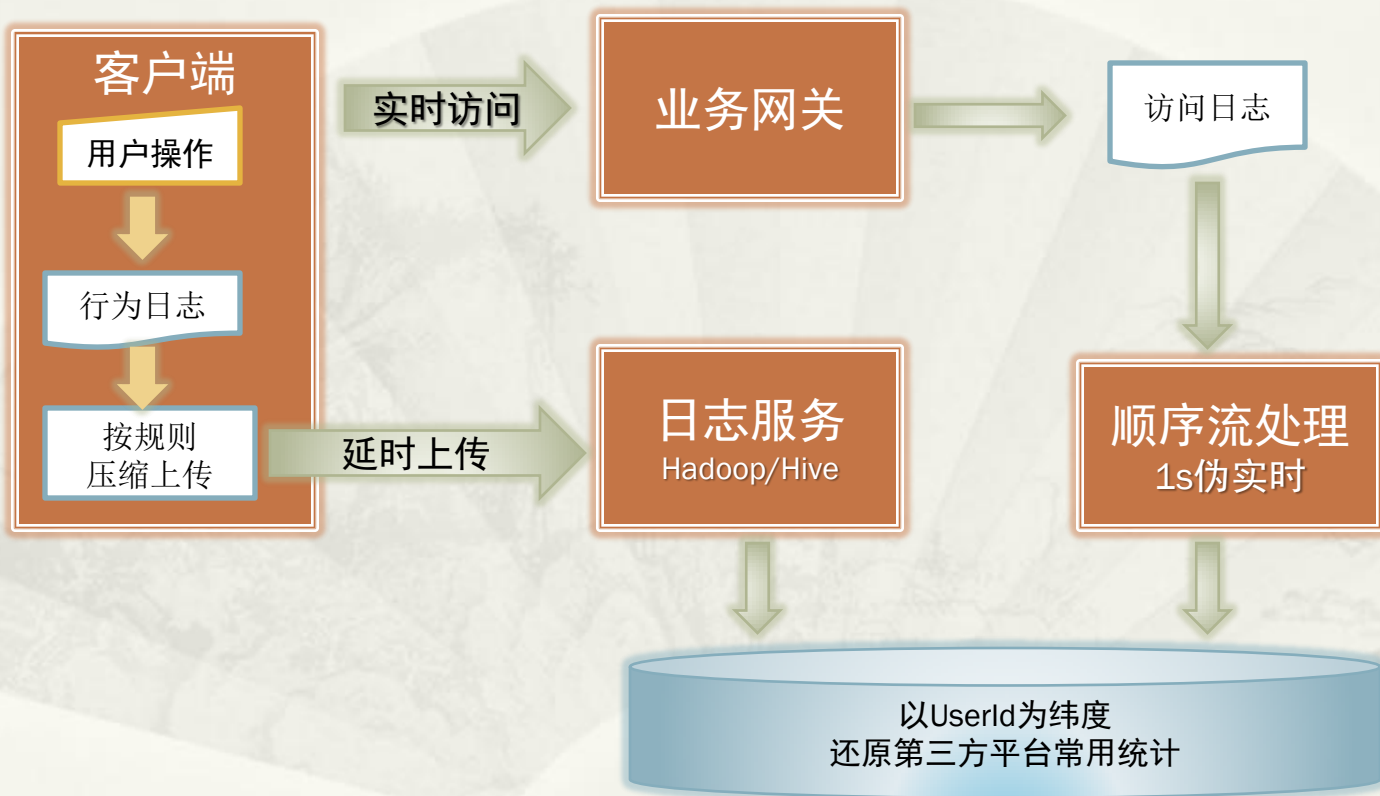
- 仅仅只有统计结果，无法跟踪到用户详细轨迹
- 仅仅统计到设备，无法统计到具体用户
- 无法做到防欺诈的安全管控，以及后续特性分析
- 敏感数据外泄

需要更加详细的数据支撑

⇓ 行为日志的诞生

- 产品需要了解各个点击、转化率
- 运营希望了解各个渠道的转化率、成本、效果
- BI希望或者各种数据的细节
- 风控希望更多基础数据支撑
- 结合第三方平台，互补分析

行为日志分析逻辑设计





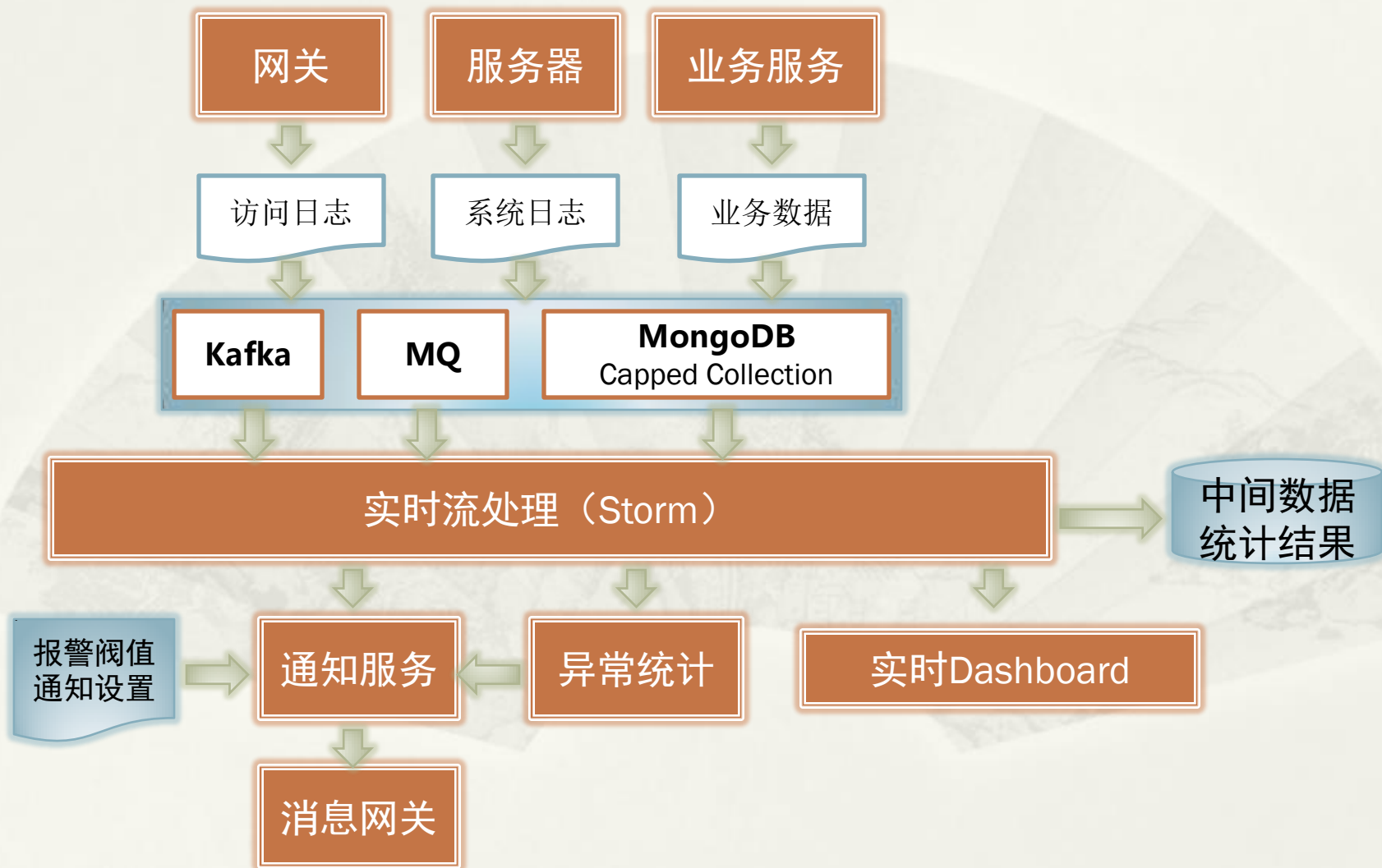
大数据 (storm)

实时数据分析改进

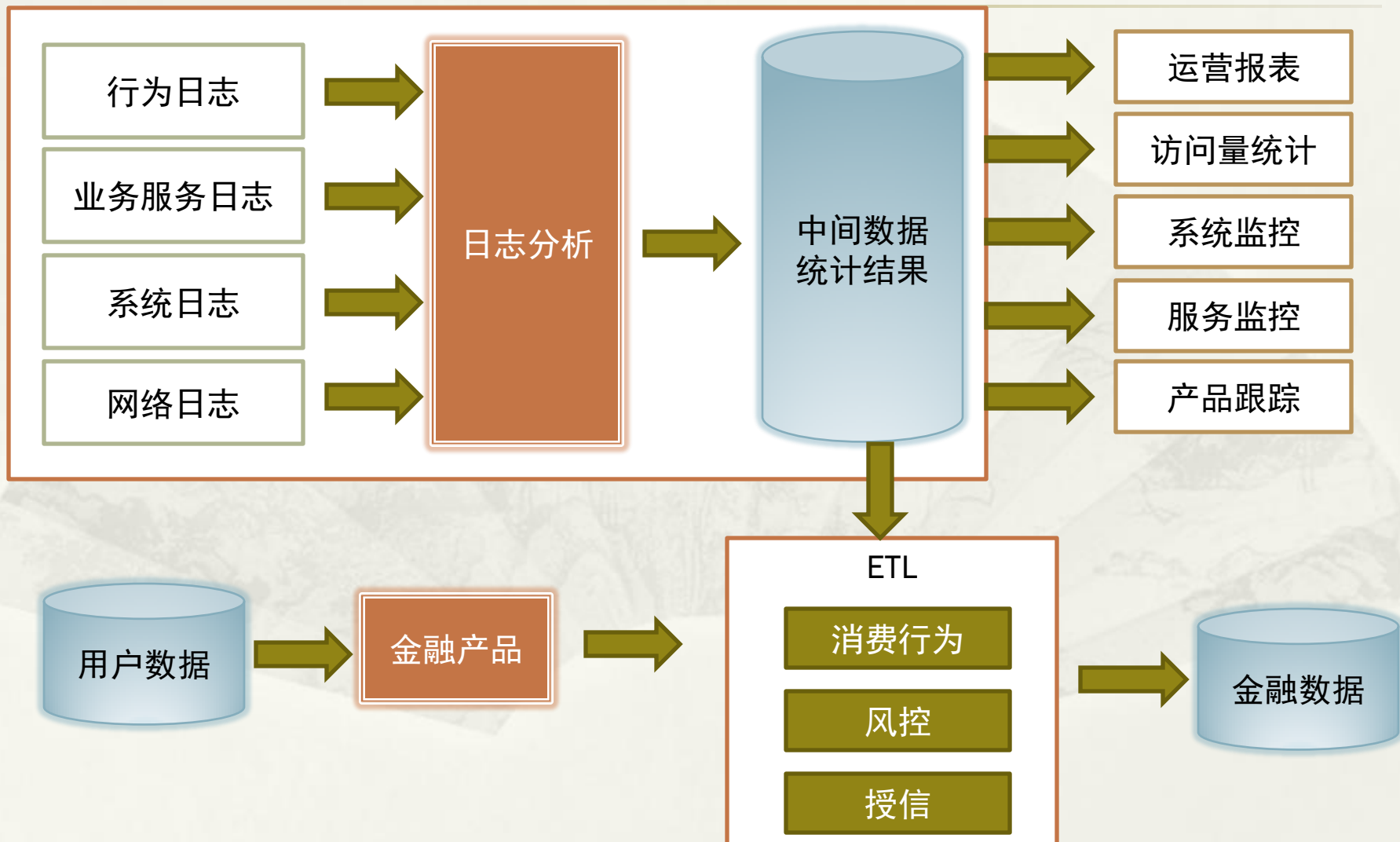
实时数据分析改进背景

- ↓ 数据量井喷时导致延迟加大
- ↓ 增加业务线需要修改代码，扩展性差
- ↓ Mongodb本身分布式能力不够，单点风险
- ↓ 分布式环境顺序流无法完全保持真正的顺序（毫秒）
- ↓ 不能根据对数据分级处理，会因某一数据过大，而影响所有分析延迟

实时数据分析改进逻辑设计



现阶段



Hadoop(hdfs/hive/map-reduce), Storm结合

数据分析平台

数据分析平台化背景

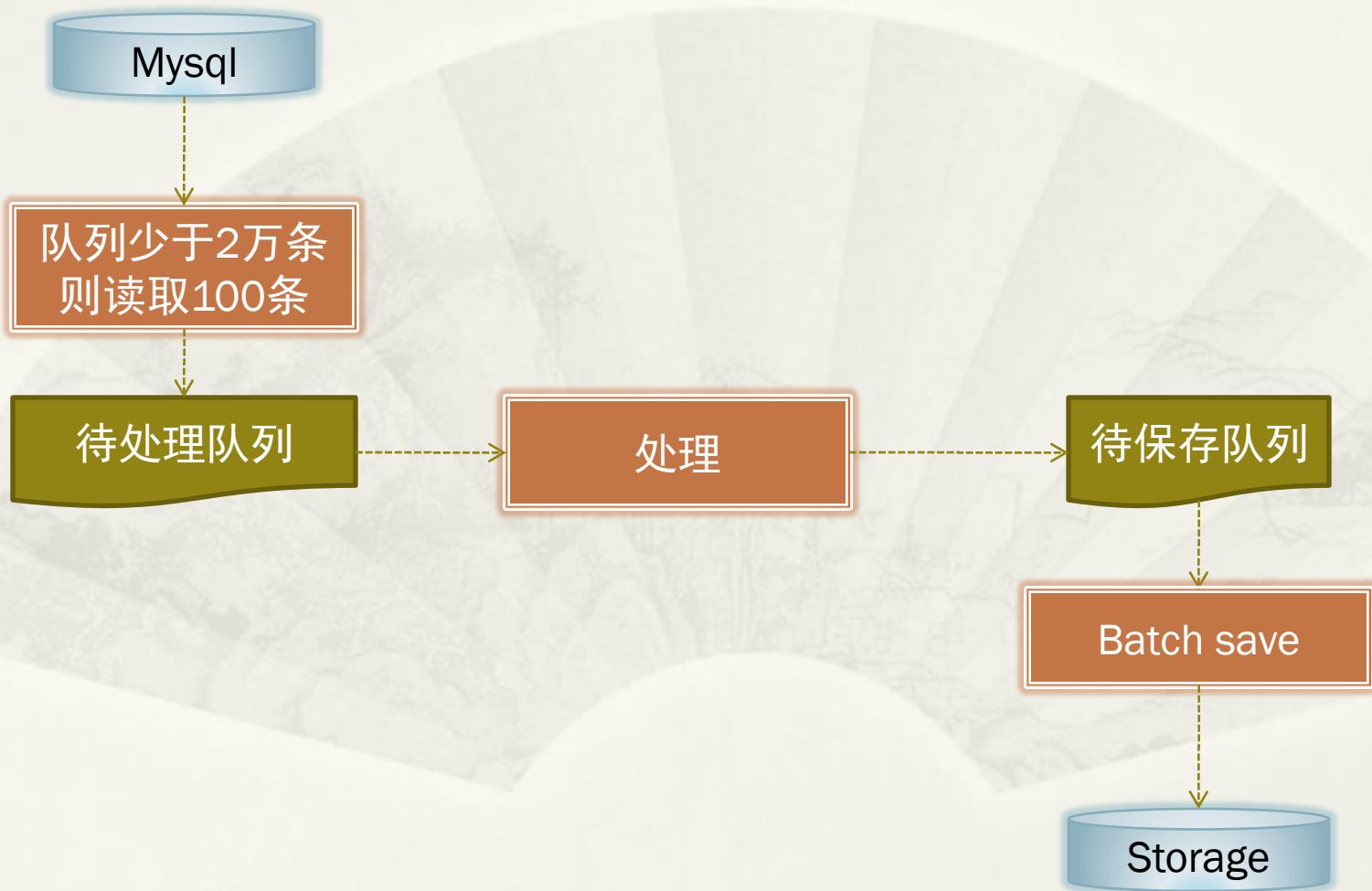
↓ 现状

- 各业务线都有特定的ETL目标，重复编写获取元数据的模块
- 各业务线都需要用到90%相同的结果，如：用户访问频次、用户注册地、用户账单分析结果
- 重复造轮子、资源浪费、入口不统一

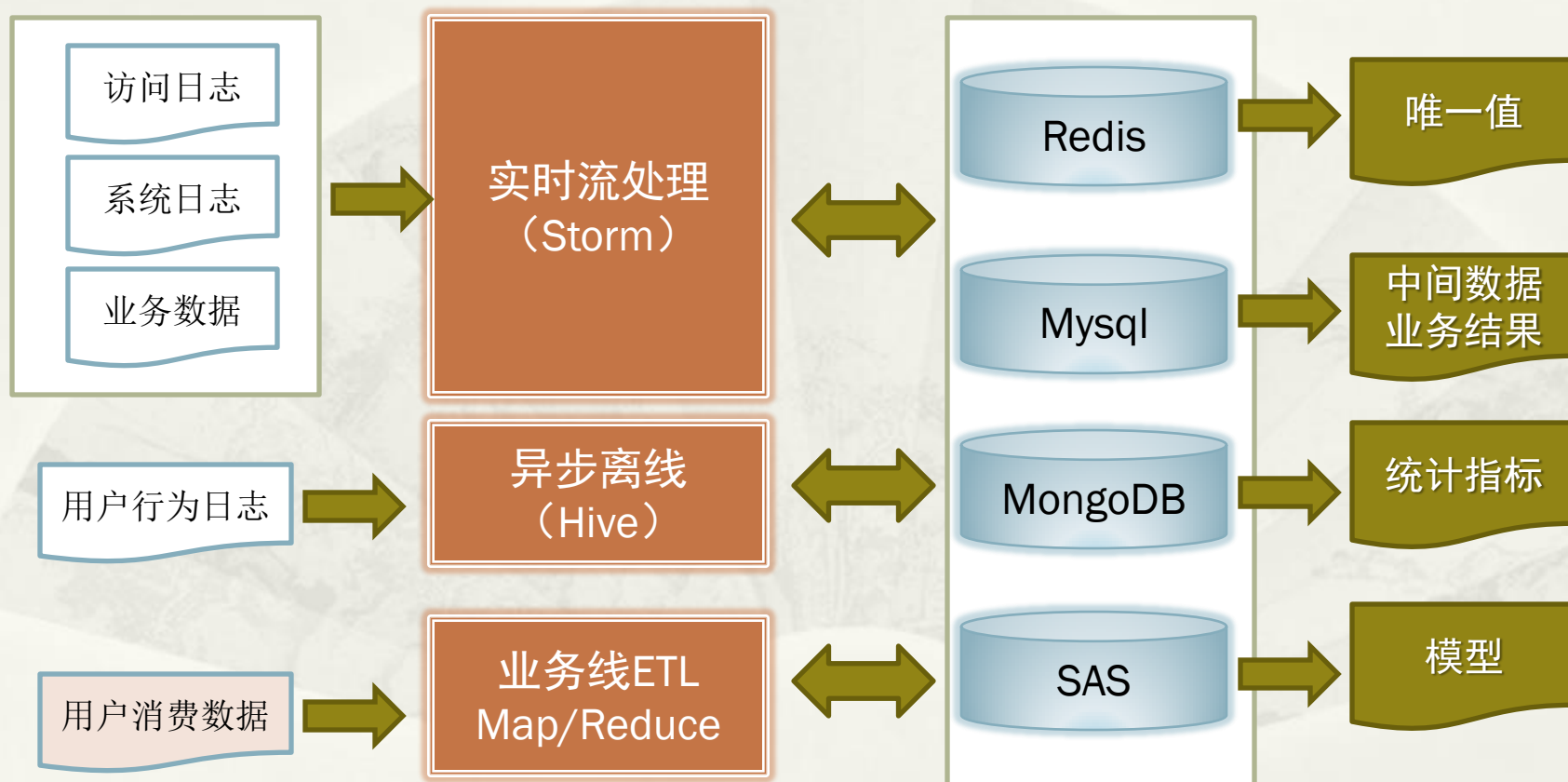
↓ 目标

- ✓ 统一的数据接口
- ✓ 统一分析过程
- ✓ 标准化IPO模式
- ✓ 只需要实现ETL逻辑

一个ETL重复造轮子的案例



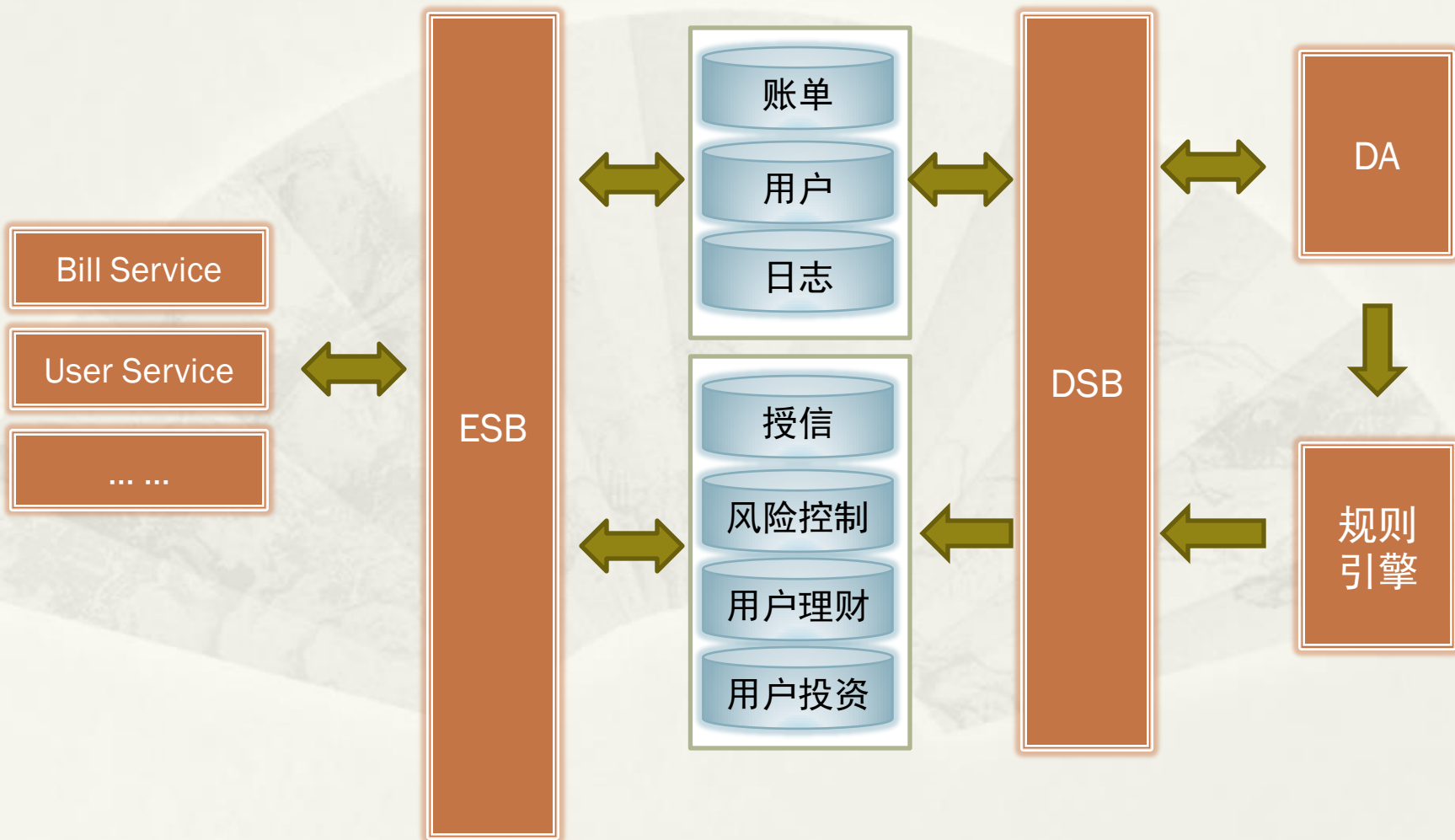
数据分析平台化




所有数据的计算规则可以灵活定制

引入规则引擎

引入规则引擎





3Q