ArchSummit全球架构师峰会深圳站2016

蘑菇街背后系统稳定性保障实践

About Me

普通程序猿
12年-至今 蘑菇街，花名苏武
经历数次蘑菇街系统改造，多为打杂
目前专注于系统稳定性相关工作

提纲

- 往年大促遇到的问题
- 问题的总结和思考
- 新思路下双11稳定性备战流程
- 总结

往年大促遇到的问题

CASE1: 13年双11当天14：20，一个活动sql导致蘑菇街主数据库慢sql急剧增加，前端访问hang住，交易下跌80%1个小时

CASE2: 14年双11大促23：50后，访问量过大雪崩，全站不可访问15分钟

CASE3: 15年321大促零点使用优惠券，更新使用数导致数据库行级锁问题，下单下跌80%10分钟

CASE4: 15年321大促零点下单数据库写入量过大，导致数据库hang住，下单下跌100%15分钟

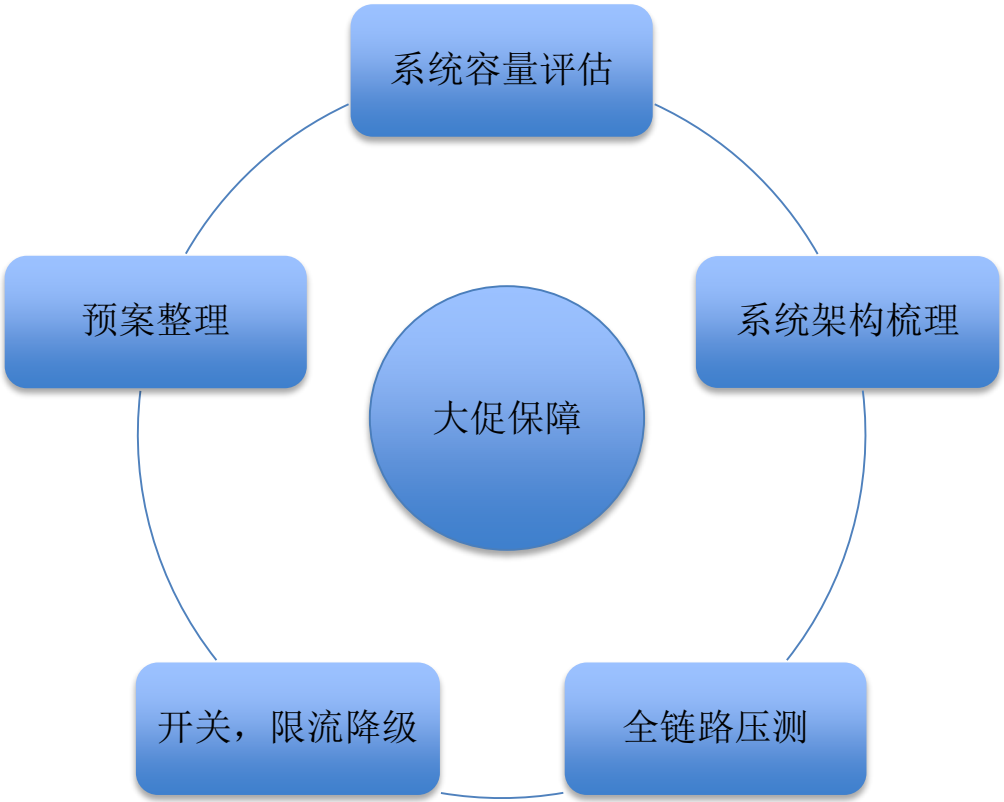
问题的总结

- 当前的系统架构应对大促有风险(CASE1，CASE3)
- 当前的系统相互之间的依赖和调用关系不清楚(CASE1，CASE3)
- 对大促系统峰值没有有效的评估(CASE2，CASE4)
- 对系统能支撑多少峰值未知(CASE2)
- 真出了问题，不能短时间内有效的应对
- 缺少完整的稳定性保障方法论，混乱

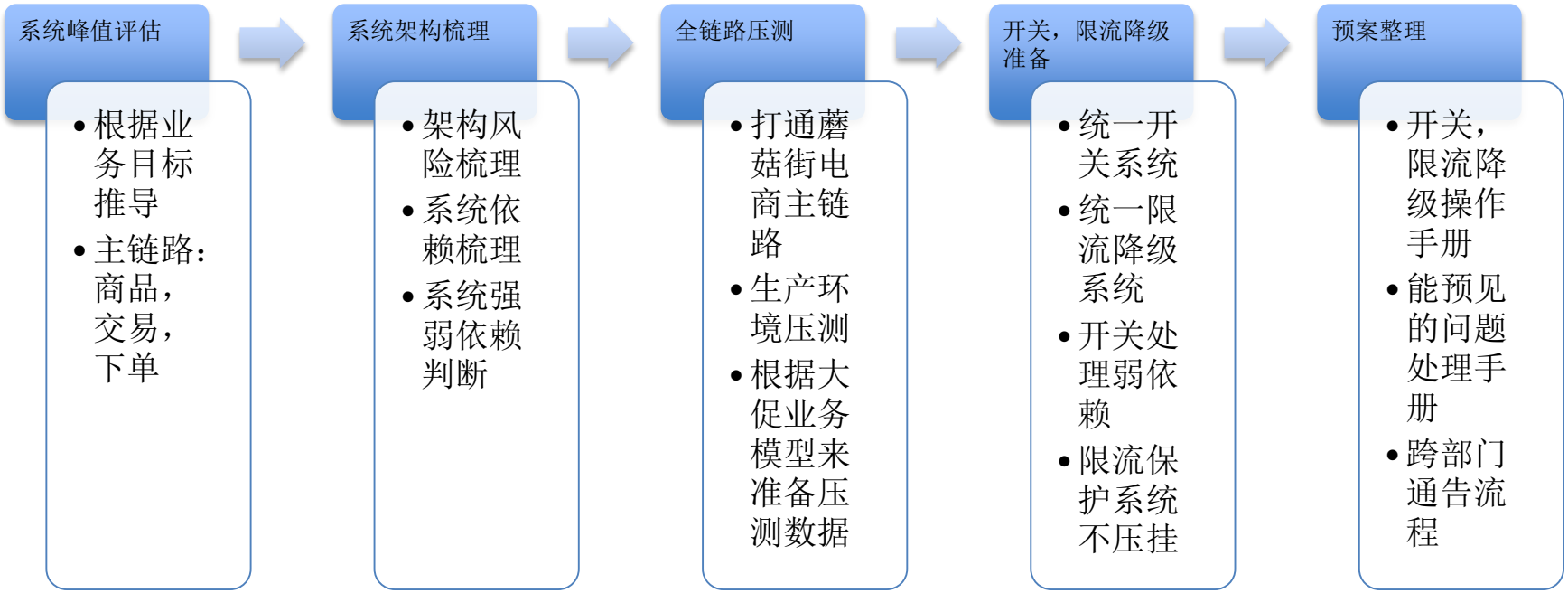
问题的思考

- 问题处理的原则
  - 解决具体问题，抽象问题后解决一类问题
  - 有沉淀，有流程，不能乱
  - 工具化，系统化代替人工操作，提高效率
  - 最终总结方法论，作为指导思想

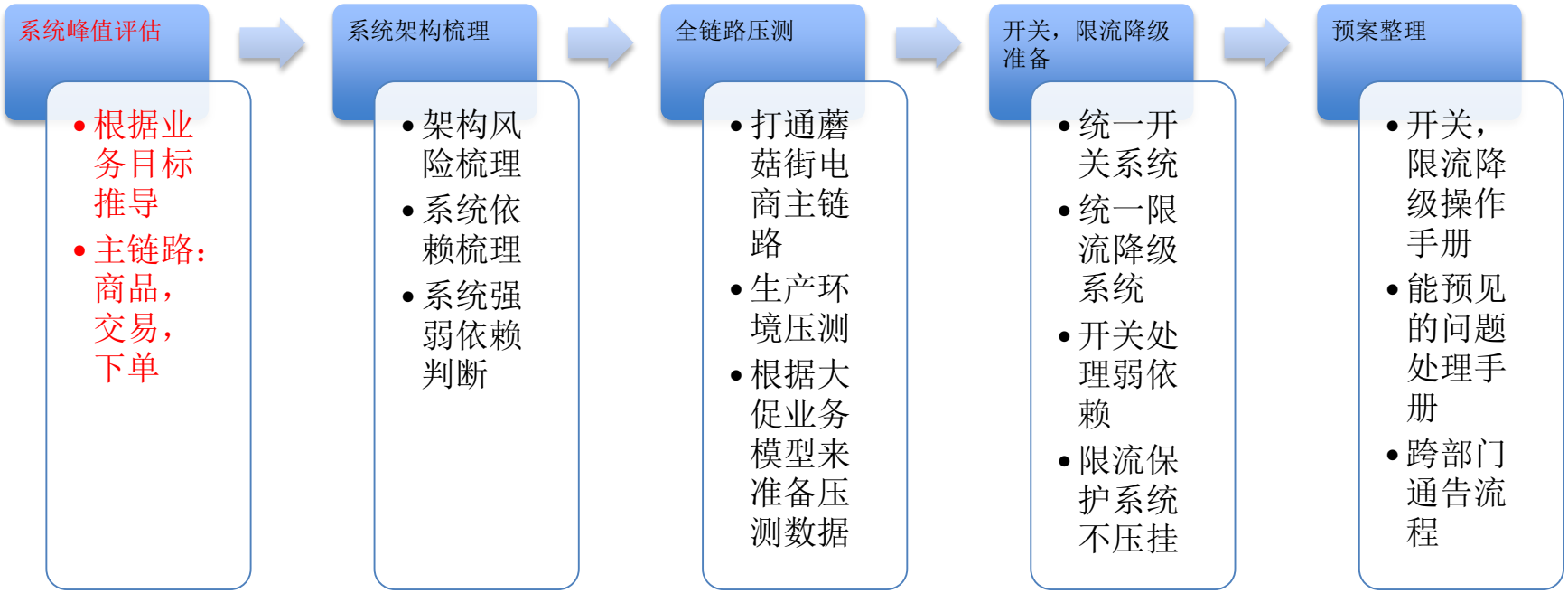
新思路下双11稳定性备战流程



新思路下双11稳定性备战流程



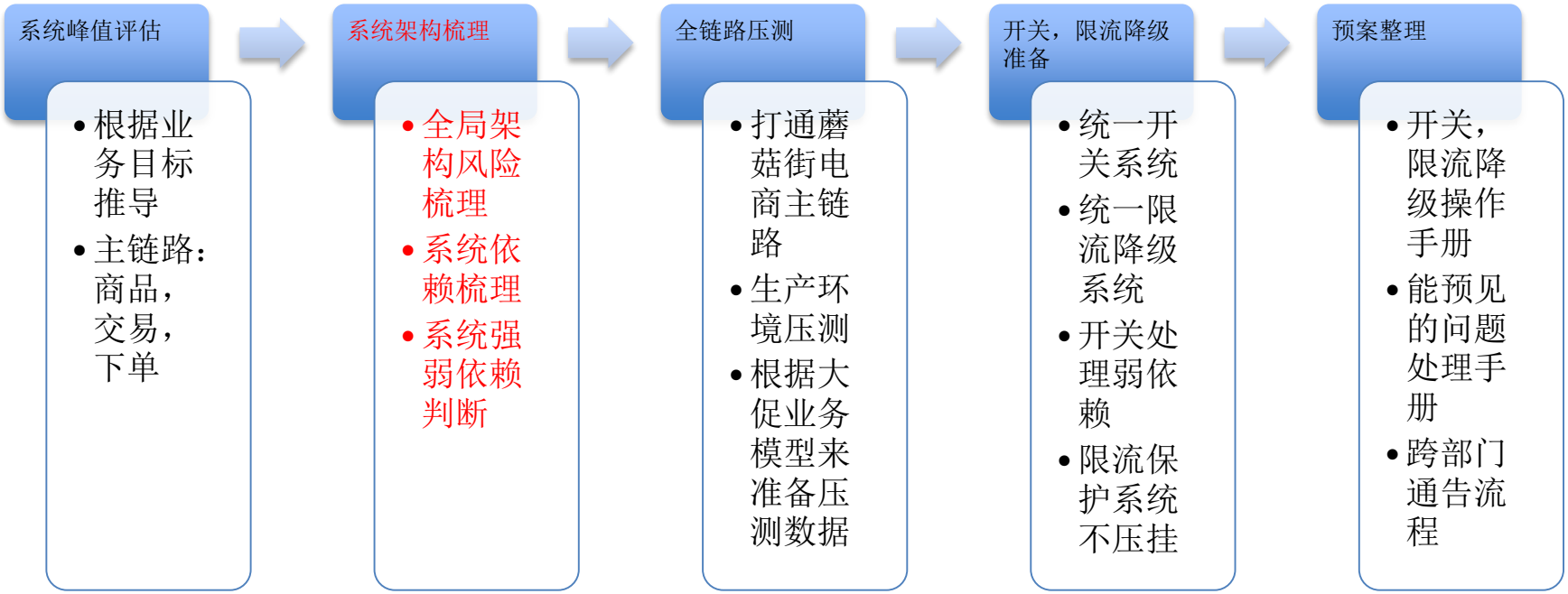
新思路下双11稳定性备战流程



系统峰值评估

- 业务目标：pv，uv，gmv，客单价，笔单价等
- 系统主要需要的：下单的峰值，支付的峰值
- 峰值推导方法1：
  - 下单峰值=天订单数/一天正常交易时段 * 大促峰值影响系数 =gmv/笔单价/8小时*8
  - 下单峰值=35,000,000/70/28800*8=140
- 峰值推导方法2：
  - 下单峰值=(本次大促gmv/上次大促gmv)*上次大促峰值*(本次大促转化率/上次大促转化率)
  - 下单峰值 =(3,000,000,000/2,000,000,000)*120*(0.30/0.25)=216

新思路下双11稳定性备战流程



系统架构梳理

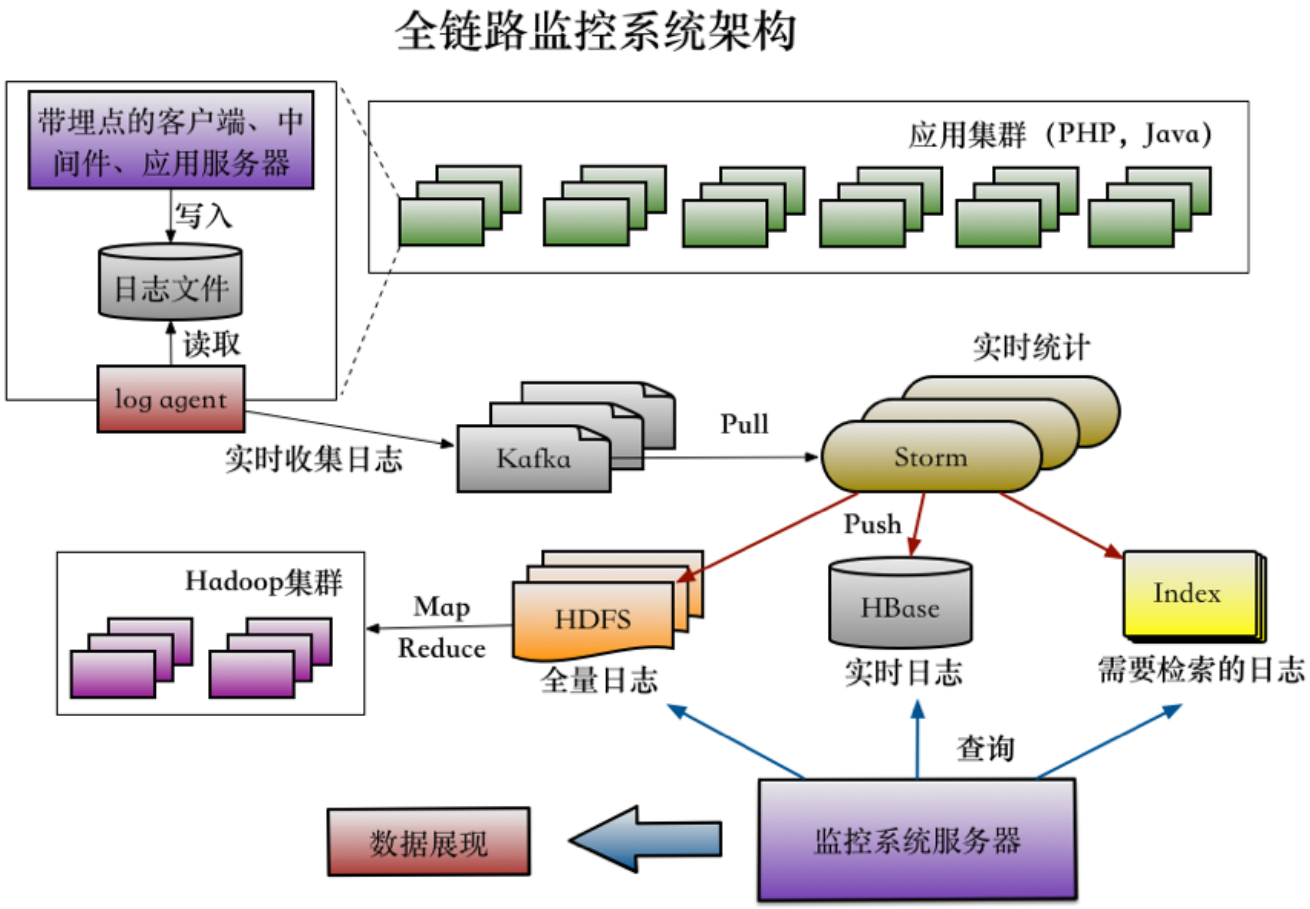
全局架构风险梳理	• 双机房流量比例确定 • 跨机房专线带宽准备 • 基础服务架构，容量梳理
系统上下游梳理	• 上下游系统依赖梳理 • 确定对上游系统调用的来源和比例，确定对下游系统的调用比例 • 标识出对下游系统的强弱依赖
系统内部梳理	• 系统内部风险梳理：如设计问题，数据热点问题 • 性能问题梳理 • 系统监控埋点梳理 • 线程池隔离梳理

系统架构梳理

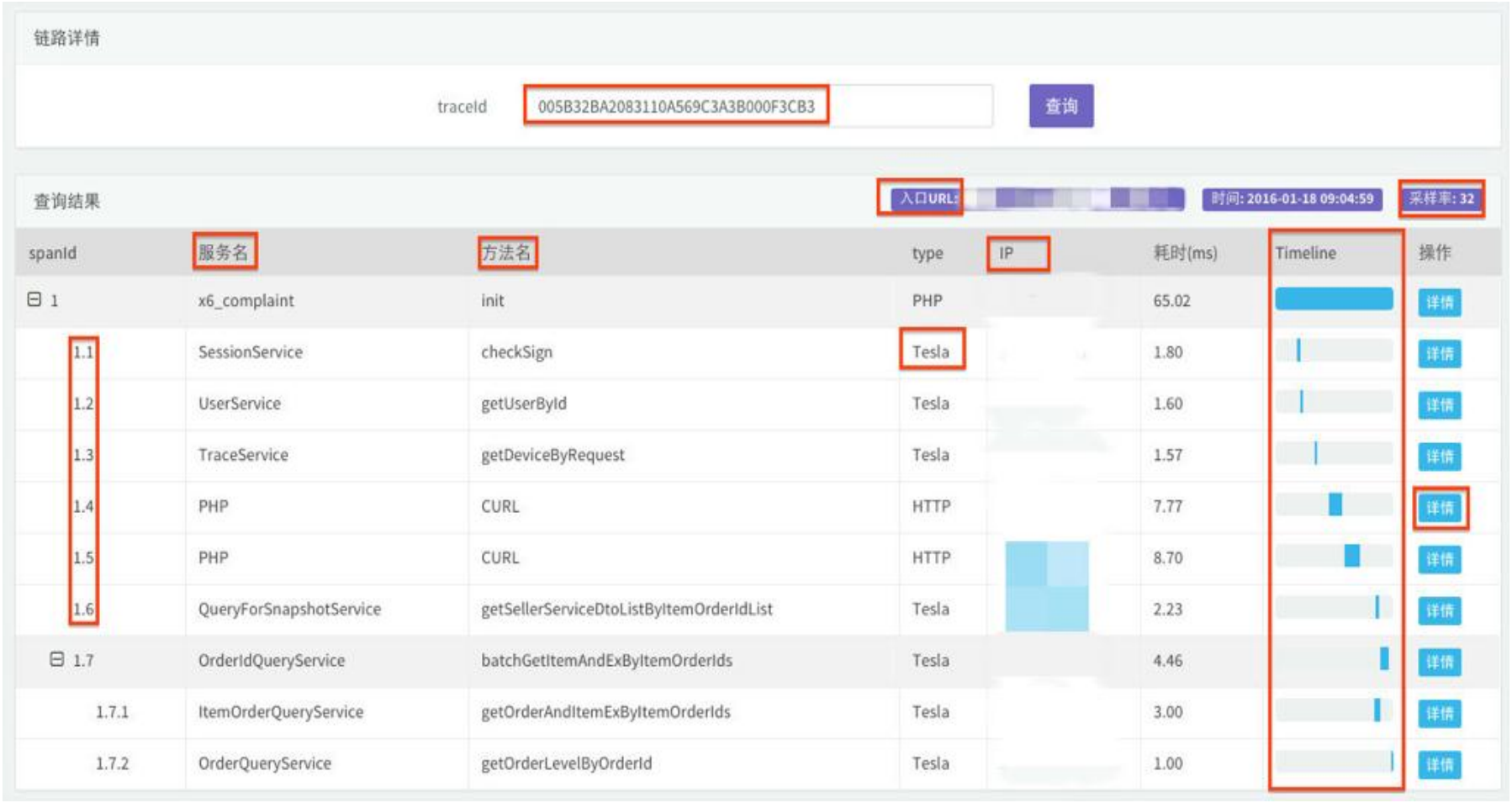
- 难点
  - 系统上下游依赖复杂
  - 系统间请求调用比例不好计算
  - 强弱依赖怎么判断，判断的标准

- 解决方案
  - 全链路监控系统
  - 强弱依赖系统，自动判断

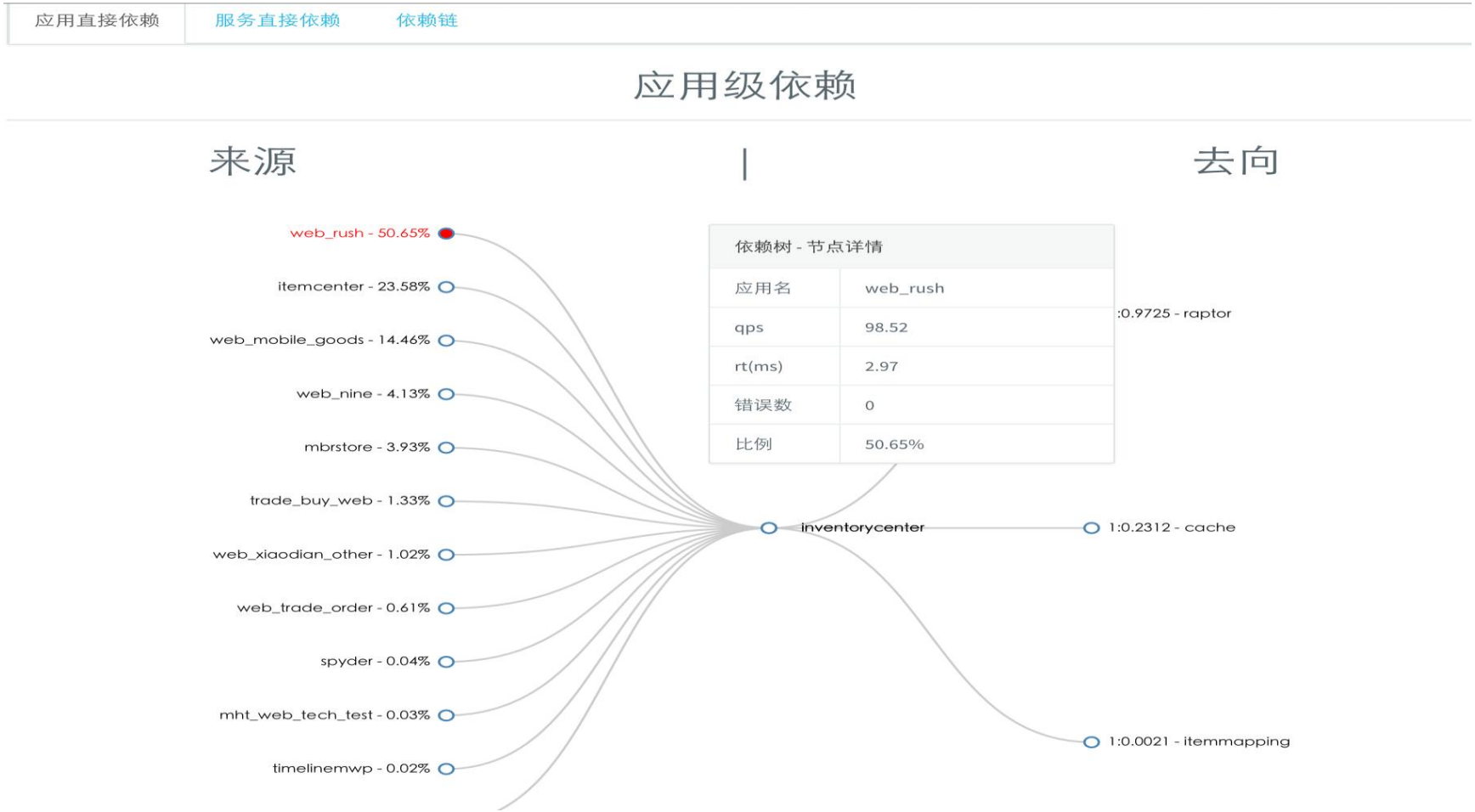
全链路监控系统


全链路监控系统架构
带埋点的客户端、中间件、应用服务器
写入
日志文件
读取
log agent
应用集群（PHP，Java）
实时收集日志
Kafka
Pull
Storm
实时统计
Push
Hadoop集群
Map
Reduce
HDFS
全量日志
HBase
实时日志
Index
需要检索的日志
查询
数据展现
监控系统服务器

单次请求追踪


链路详情
traceId
005B32BA2083110A569C3A3B000F3CB3
查询
查询结果
入口URL:
时间: 2016-01-18 09:04:59
采样率: 32
spanId
服务名
方法名
type
IP
耗时(ms)
Timeline
操作
1 x6_complaint init PHP 65.02 详情
1.1 SessionService checkSign Tesla 1.80 详情
1.2 UserService getUserById Tesla 1.60 详情
1.3 TraceService getDeviceByRequest Tesla 1.57 详情
1.4 PHP CURL HTTP 7.77 详情
1.5 PHP CURL HTTP 8.70 详情
1.6 QueryForSnapshotService getSellerServiceDtoListByItemOrderIdList Tesla 2.23 详情
1.7 OrderIdQueryService batchGetItemAndExByItemOrderIds Tesla 4.46 详情
1.7.1 ItemOrderQueryService getOrderAndItemExByItemOrderIds Tesla 3.00 详情
1.7.2 OrderQueryService getOrderLevelByOrderId Tesla 1.00 详情

依赖统计


应用直接依赖
服务直接依赖
依赖链
应用级依赖
来源
去向
web_rush - 50.65%
itemcenter - 23.58%
web_mobile_goods - 14.46%
web_nine - 4.13%
mbrstore - 3.93%
trade_buy_web - 1.33%
web_xiaodian_other - 1.02%
web_trade_order - 0.61%
spyder - 0.04%
mht_web_tech_test - 0.03%
timelinemwp - 0.02%
依赖树 - 节点详情
应用名 web_rush
qps 98.52
rt(ms) 2.97
错误数 0
比例 50.65%
inventorycenter
:0.9725 - raptor
1:0.2312 - cache
1:0.0021 - itemmapping

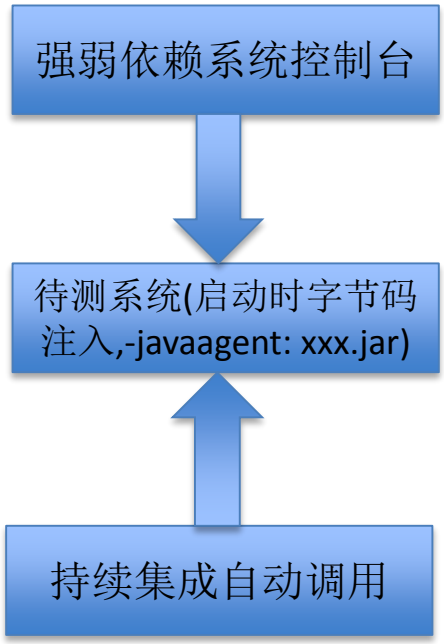
实现细节

- TraceID：全局唯一，接收请求最上层生成，透传到底；组成： ip+进程id+随机数+秒+微妙
- SpanID：一个调用链下的多个调用的发生顺序和嵌套层次关系
- TraceID和SpanID传递，低侵入：nginx module；php 扩展；java中间件(ThreadLocal透传)
- java异步调用， ThreadLocal无效，解决方案：
  - 业务代码自己处理和传递
  - 提供定制的ThreadPoolExecutor，重写beforeExecute ,afterExecute进行传递和清除
- java应用写日志的压力，解决方案：
  - 自己实现异步写日志
  - 写内存，按时间或者大小刷盘，允许丢

强弱依赖

- 定义：当一个依赖调用出现问题，可以暂时屏蔽或者可以调用备用依赖来提供有损服务的就是弱依赖，反之则是强依赖
- 弱依赖出现问题的处理方式
  - 推送开关直接屏蔽
  - 推送开关切到备用依赖
  - 配置降级策略，在一段时间内调用返回失败，前端应用做友好提示
- 强依赖出现问题的处理方式
  - 强依赖原则上不能出问题，在准备阶段重点梳理，多次压测
  - 系统设计上错误的将弱依赖变成了强依赖，需要系统改造
  - 强依赖必须配置限流保护
  - 理论上还会出现问题，必须有针对强依赖出现问题的预案

强弱依赖系统



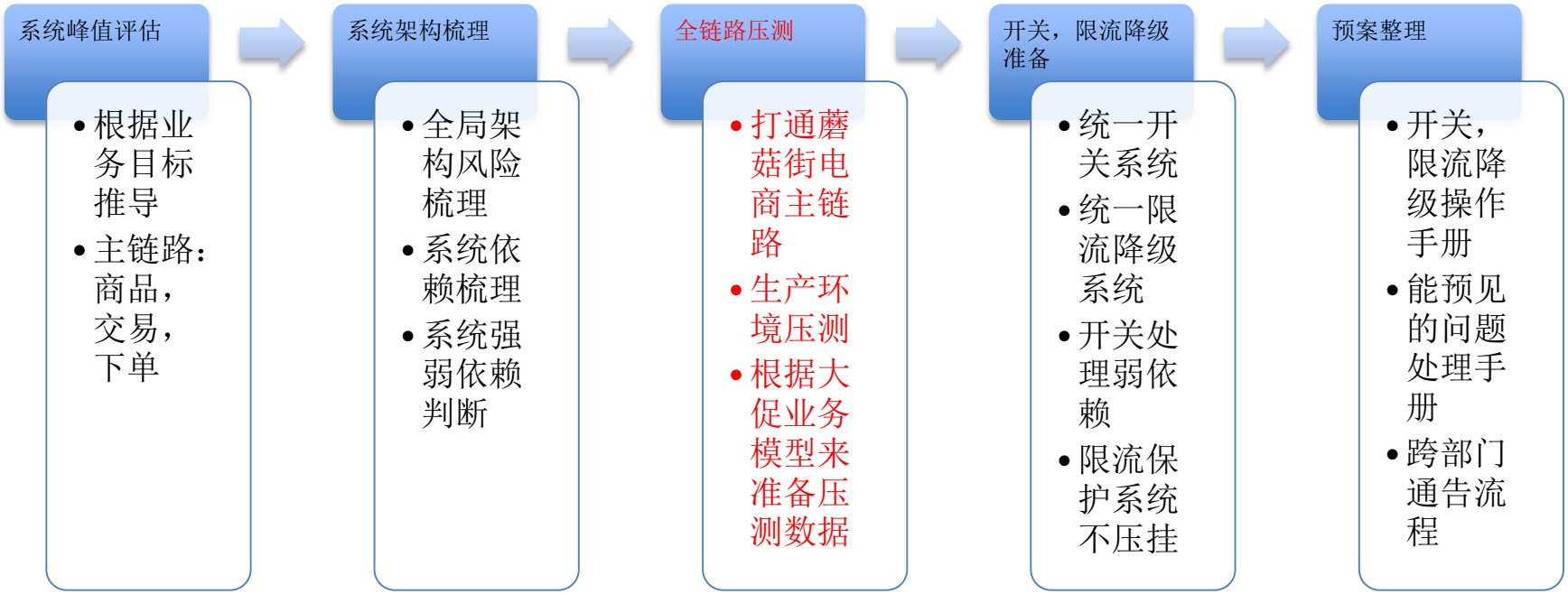
开关控制是否开启测试
场景：异常，返回值错误，网络延时，网络屏蔽，开关，降级

```
public void testvoid() {
    BeforeMethodHander beforeMethodHander = new BeforeMethodHander(3);
    if (beforeMethodHander.isSwitchOpen()) {
        Strategy localStrategy = beforeMethodHander.doHandler();
        localStrategy.invoke();
    }
    /**
     * 业务代码
     */
}
```

注入的代码

持续集成根据控制台的配置进行自动调用，每日产出报表

新思路下双11稳定性备战流程



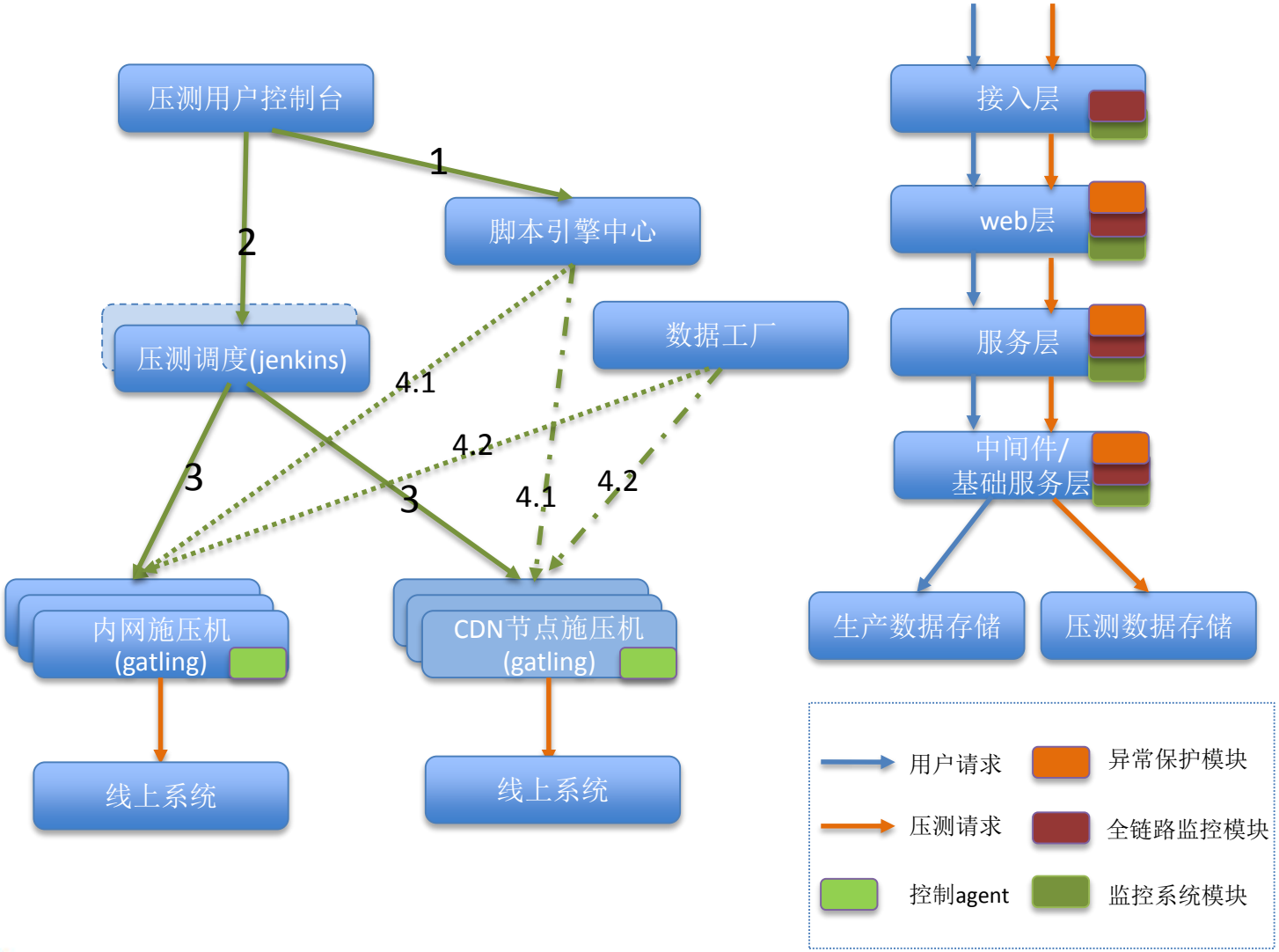
全链路压测

- 传统的压测方式
  - 线下压测，线上压测
  - 引流压测，日志回放压测，造数据压测，流量放大压测
- 问题
  - 线下压测压不准/线上压测很危险
  - 关注点在单机或者单集群，多个系统串联很难
  - 写操作/有状态系统压测困难

全链路压测

- 线上真实环境压测
- 模拟用户真实请求，全站性质压测
- 模拟大促特点导致的特殊的系统行为
- 可以根据业务模型调整压测模型
- 处理压测带来的脏数据
- 压测不会影响用户的正常请求

全链路压测


压测用户控制台
1
脚本引擎中心
2
压测调度(jenkins)
数据工厂
4.1
4.2
3
3
4.1
4.2
内网施压机
(gatling)
CDN节点施压机
(gatling)
线上系统
线上系统
接入层
web层
服务层
中间件/
基础服务层
生产数据存储
压测数据存储
用户请求
压测请求
控制agent
异常保护模块
全链路监控模块
监控系统模块

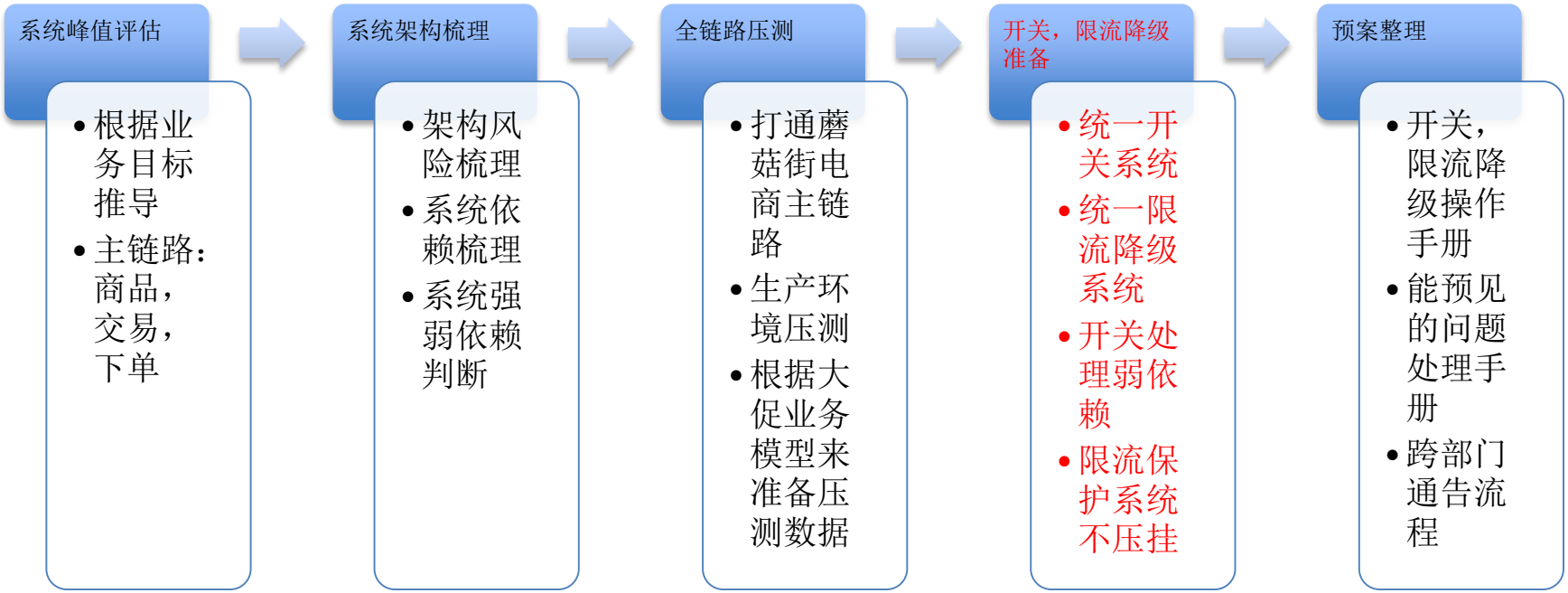
全链路压测

- 线上系统端
  - 压测标透传，通过全链路监控系统，中间件/基础服务改造
  - 影子存储，压测标判断(+业务上压测请求数据判断)
  - 异常保护模块，保护线上请求，熔断压测请求
  - 监控模块，提供数据给压测端Agent做异常控制

全链路压测

- 压测系统端
  - 脚本引擎，固化通用部分，控制台输入变化的部分，引擎产生压测脚本
  - 数据工厂，压测数据输入/输出源
  - 压测链路，一个或者多个压测请求组合
  - 压测场景，对压测链路的复杂组合
  - 异常控制，在线上系统异常的情况下停止压测

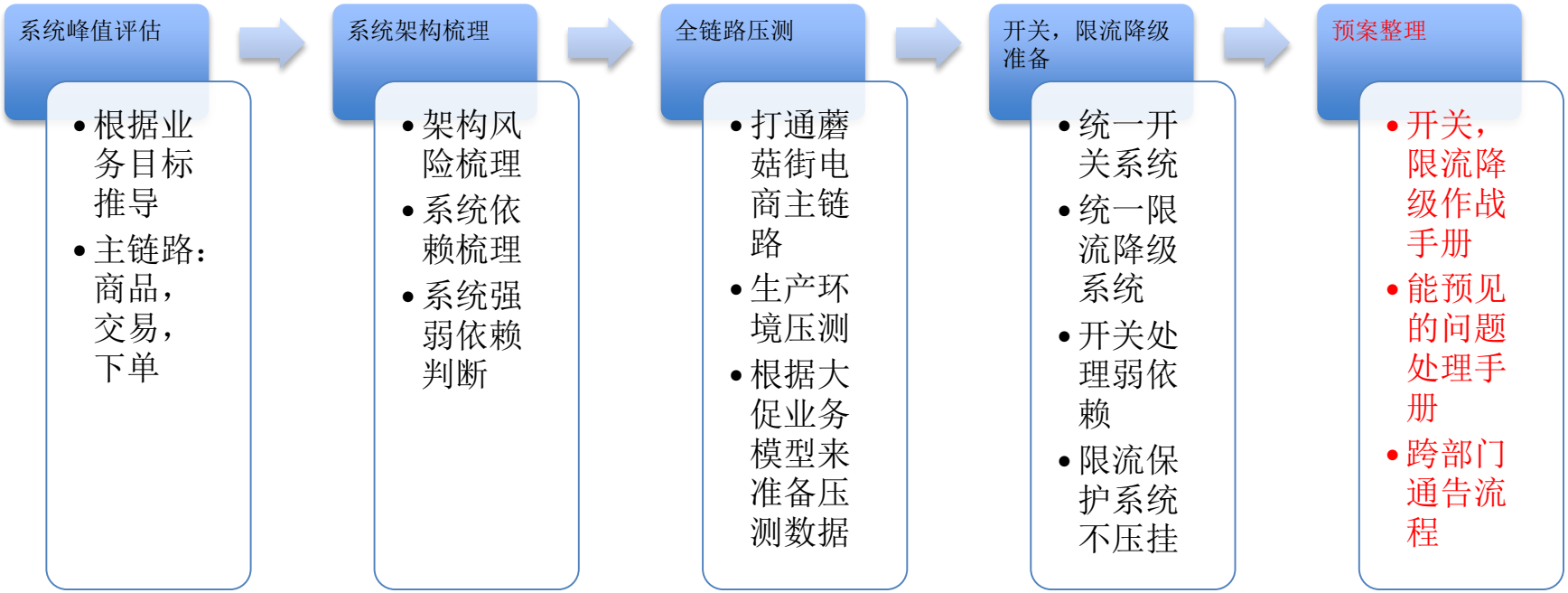
新思路下双11稳定性备战流程



开关，限流降级

- 开关
  - 对弱依赖的处理
  - 统一的开关推送系统，统一的开关监控
  - 多种推送方式，确保可以推送成功(配置中心，http)
- 限流降级
  - 保护系统不会被压垮
  - 限流类型：QPS/并发,来源/去向
  - 入口层进行QPS的限流
  - 系统内部下游对上游QPS限流，上游对下游并发限流
  - 降级是对弱依赖不可用做的措施，自动在一段时间窗口内屏蔽该依赖

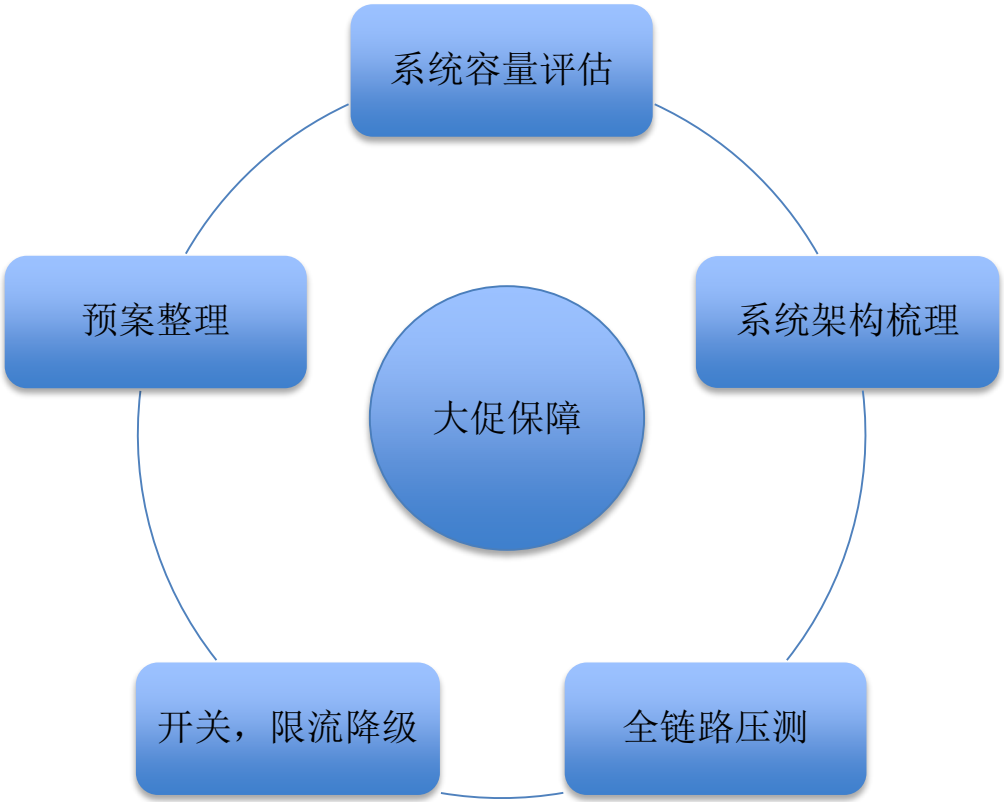
新思路下双11稳定性备战流程



预案整理

- 开关和限流降级手册
- 能预见的所有问题的处理手册；如：某个机房不可用的整体预案
- 跨部门通告流程
- CASE：某个DB的master挂掉

总结



Thanks!