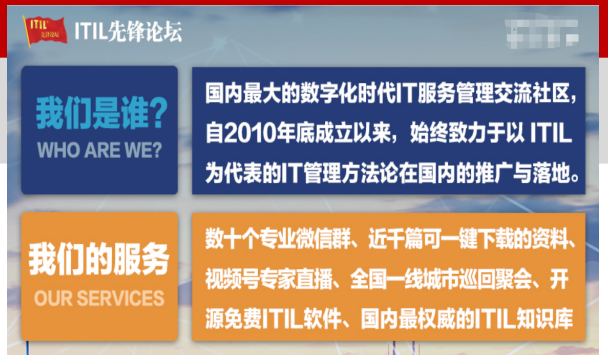
双体系：XX金融运维平台实践

运维平台负责人



GOPS 2020
shenzhen

CONTENTS

目录

01

项目背景

---金融与互联网结合对运维的挑战

业务背景和现状

海量实时业务7*24小时永动机

1、业务海量
十万+/秒支付、百万+/秒入账
十亿+/天

2、变更频繁
在奔跑的高速上换轮子

用户要求

可用：>99.999%

快：<200ms

安全：资金有保障

没有什么是确定的

硬件不可靠

程序有bug

人会犯错误

对运维平台的要求是？

02

整体解决方案

---全方位兜住不确定性，降低未知风险的影响

定义最核心问题

GOPS 2020
shenzhen

问题一	现网频繁变更，如何不人为搞出故障？
问题二	故障不可避免，出现故障，如何快速恢复业务？
问题三	业务现在没问题，提前解决那些风险隐患导致业务未来有故障的问题？

整体解决方案

GOPS 2020
shenzhen

统一变更

核心目标：
确保变更对业务可用的影响可控

故障处理

核心目标：
减少故障对业务可用的影响时间

持续运营

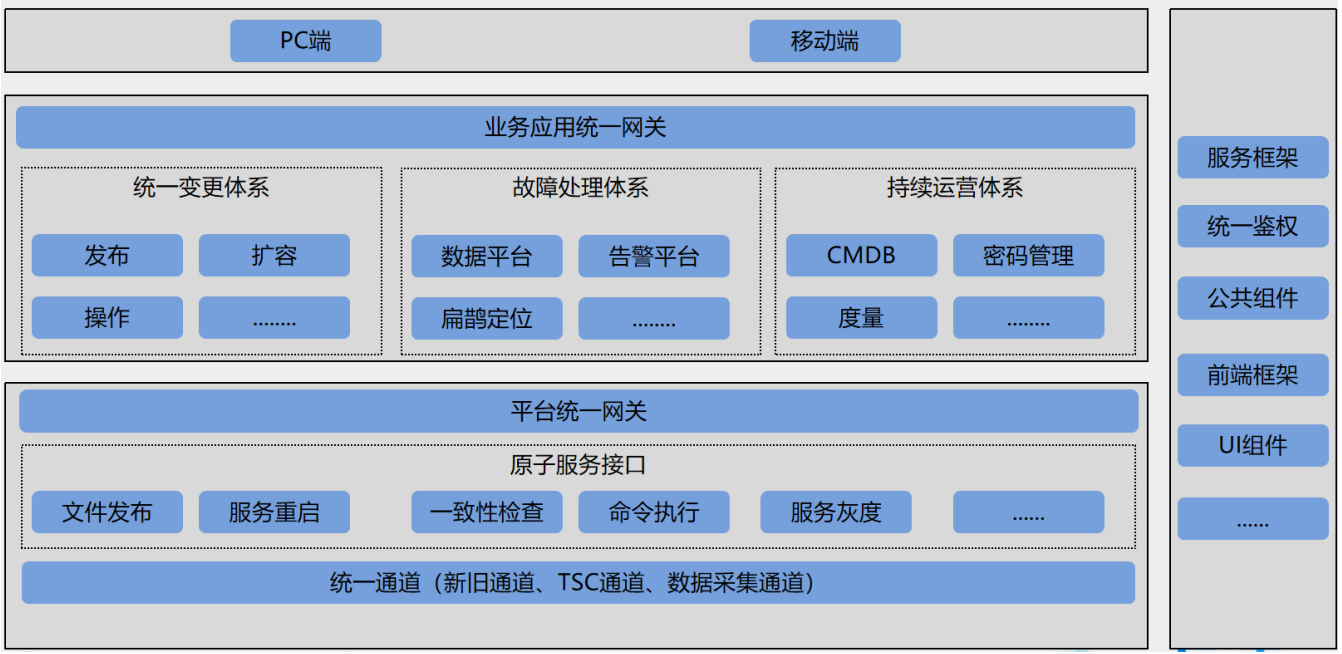
核心目标：
持续减少业务可用的隐患

一切都是为可用性

整体解决方案

磐石

FIT品质　　坚如磐石

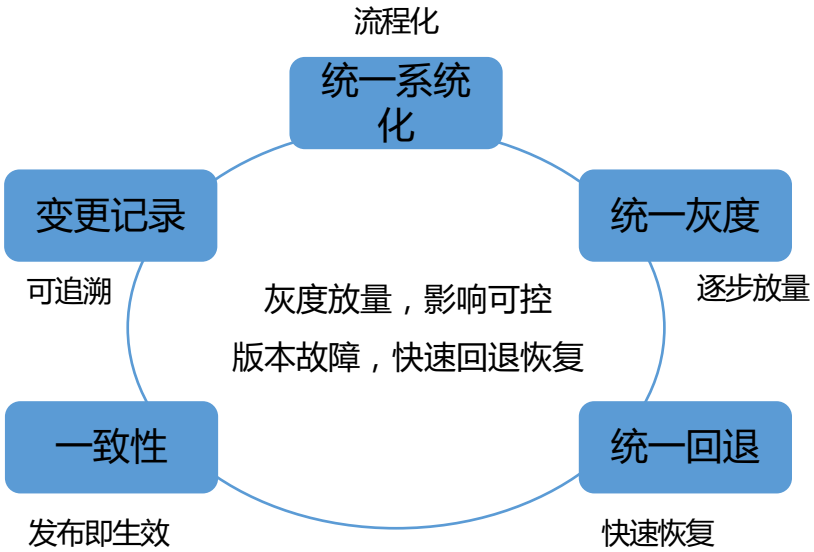


03

统一变更体系

---业务无损变更方案，控制变更时不搞出故障

统一变更：整体方案以及难点



难点：

1、确保变更对现网影响可控？

2、和现网一样级别的可用性？

统一变更：具备切换能力的灰度发布规则

	业务1		业务2	
灰度1	城市 10%	对等城市	城市 1台	对等城市
灰度2	城市	对等 10%	城市	对等 10%
灰度3	城市 30%	城市	城市 10%	对等城市
灰度4	城市	对等 30%	城市	对等 30%
灰度5	城市 100%	对等城市	城市 30%	对等城市
灰度6	城市	对等 100%	城市	对等 100%
全量			城市 100%	对等城市

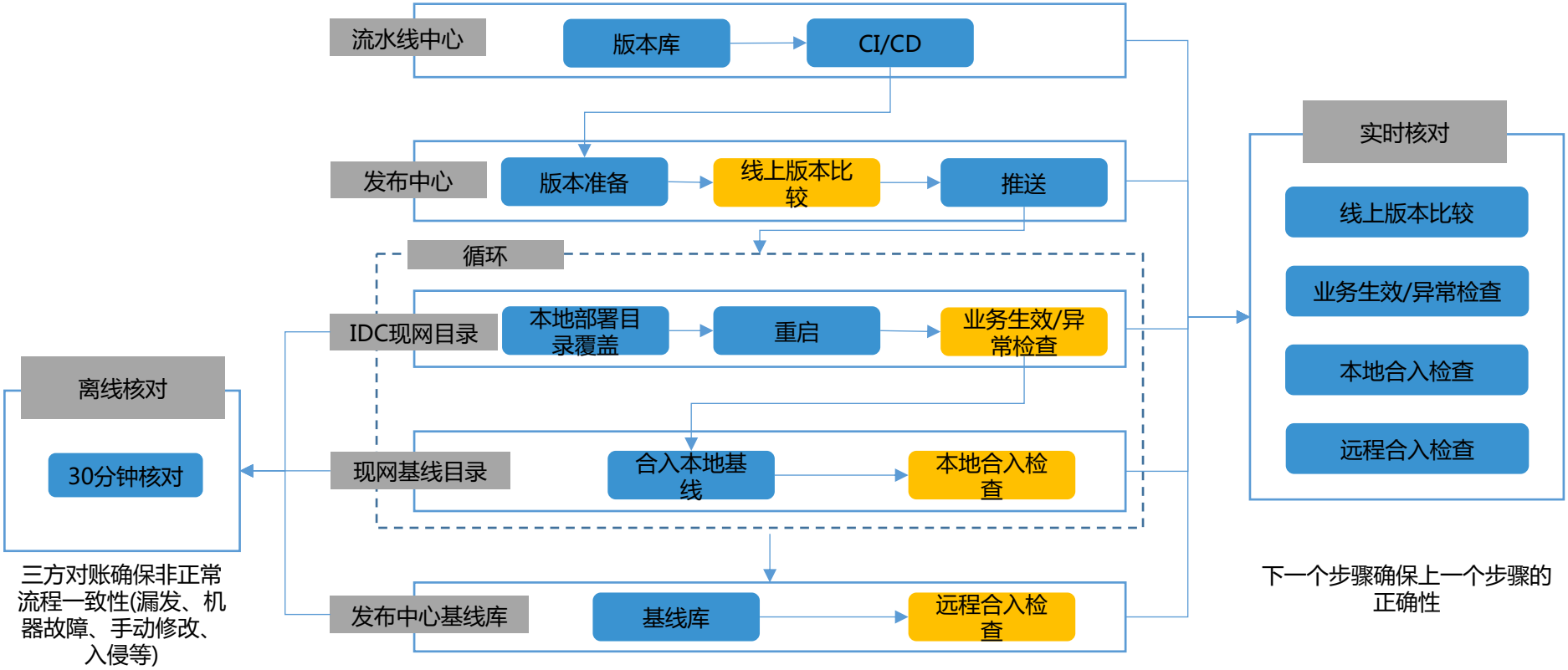
灰度原则能力：

1、业务按照优先级
2、流量逐步放量
3、一个步骤内只动单边

变更中兜底能力：

1、流量切换到对端(自动/手动)
2、版本回退
3、版本基线回退(很少用)

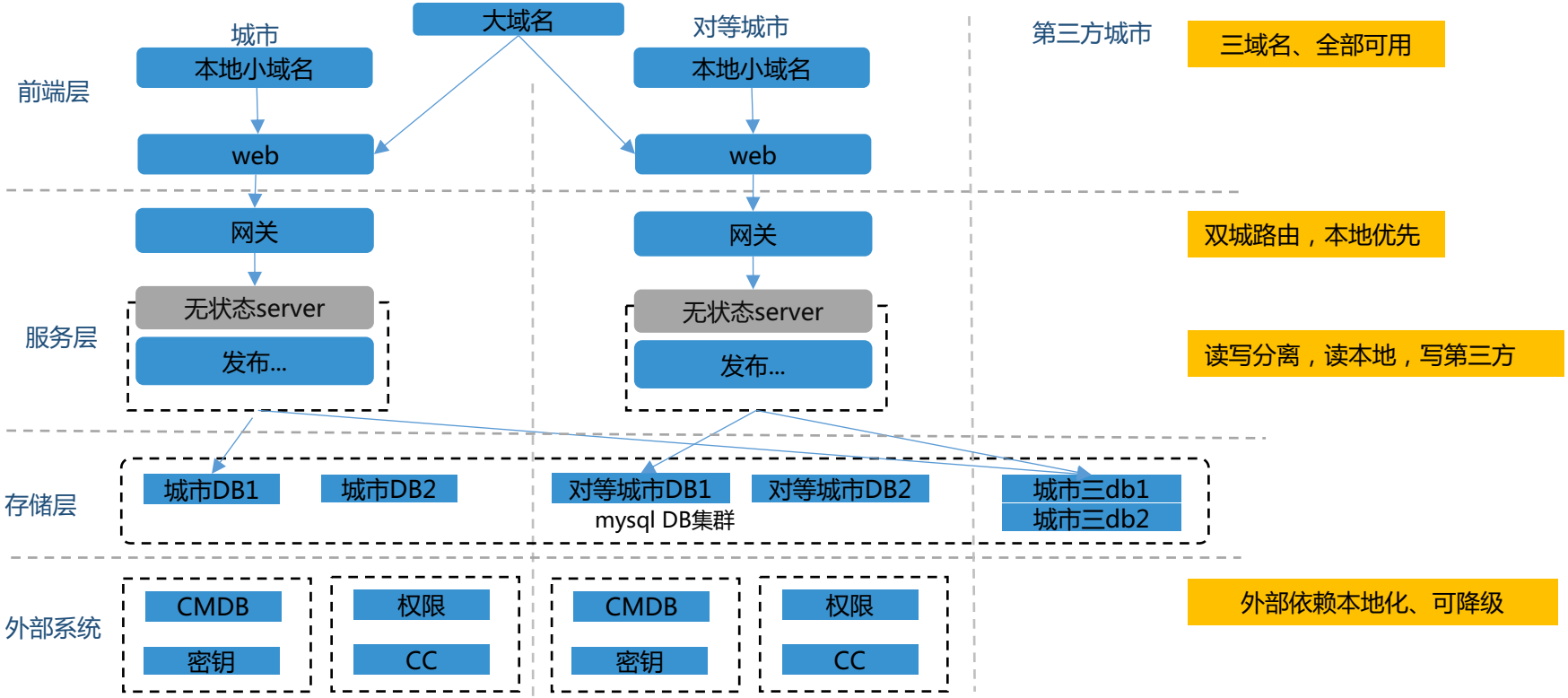
统一变更：发布即生效一致性解决方案



三方对账确保非正常流程一致性(漏发、机器故障、手动修改、入侵等)

下一个步骤确保上一个步骤的正确性

统一变更：双城双活的高可用发布平台



统一变更体系小结

统一变更目标：变更发生时，确保变更过程“安全可控”，减少对可用度的影响。

发布过程可控

不留坑给下一次

大故障救命急用

04

故障处理体系

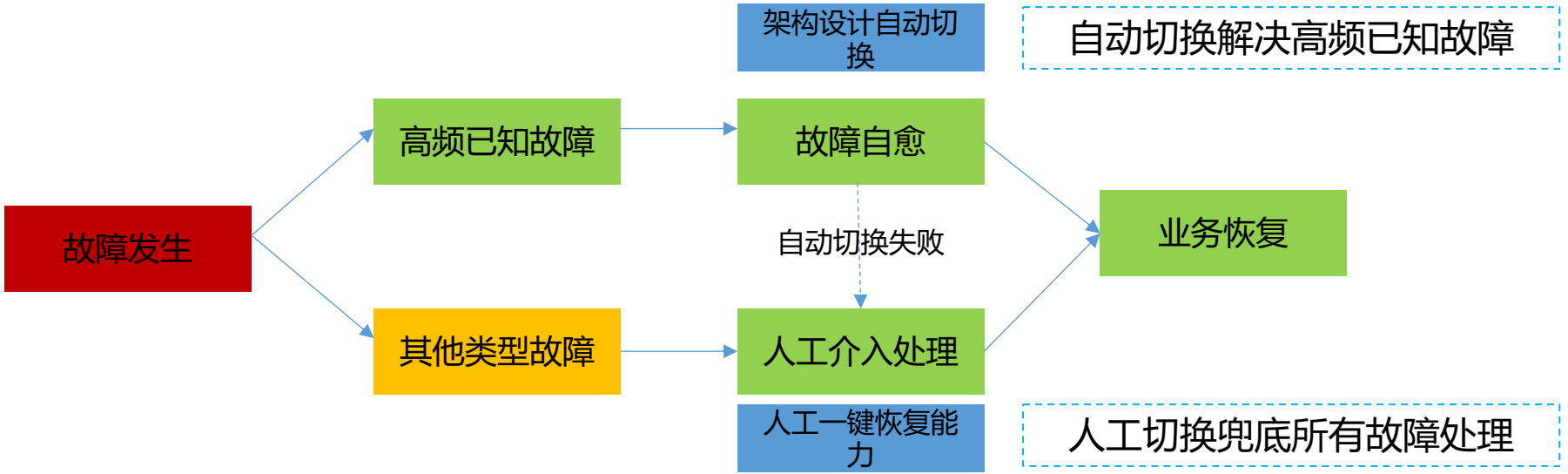
---快速恢复故障，降低业务影响

故障处理整体思路

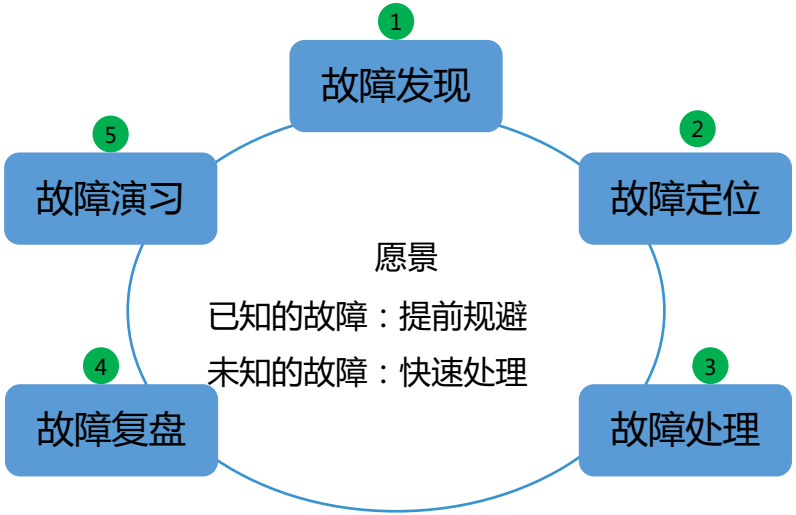
目标

快

最快速处理故障方式恢复业务，减少业务影响。



故障处理：整体方案及难点



难点：

1、海量数据又快又准？

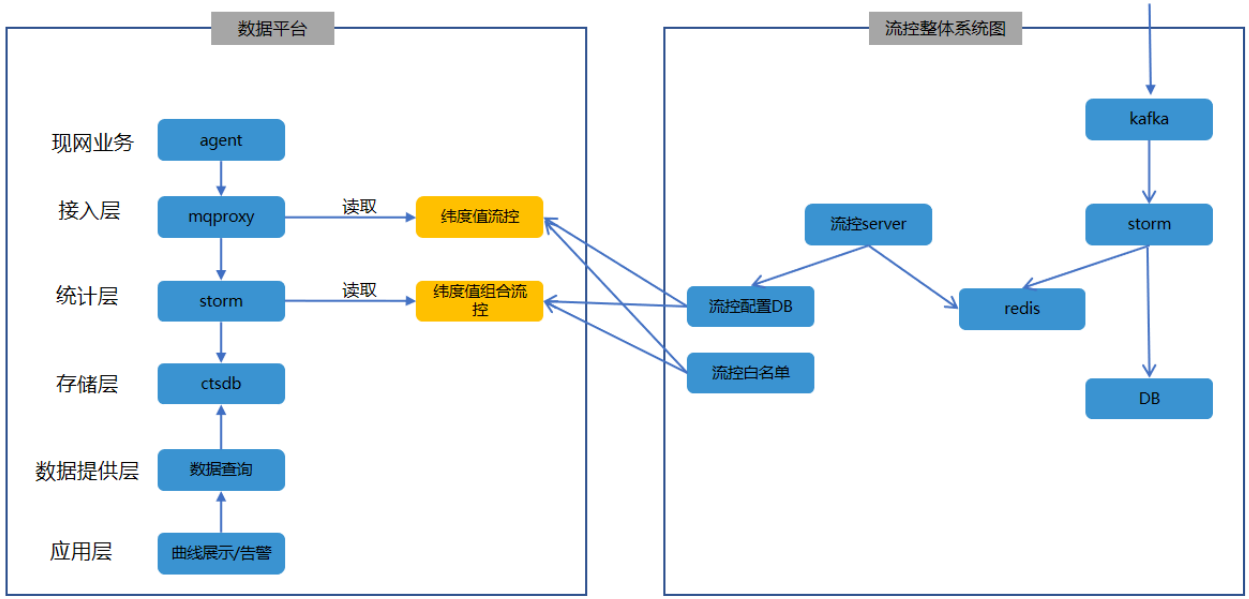
2、故障发现(告警)又快又准？

3、故障定位怎么做准？

4、故障处理简单有效？

故障处理：纬度值泛滥的流控方案

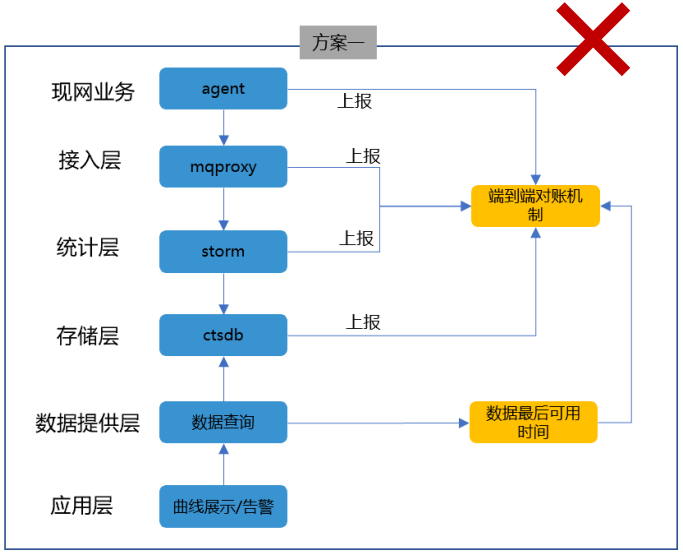
纬度值泛滥？



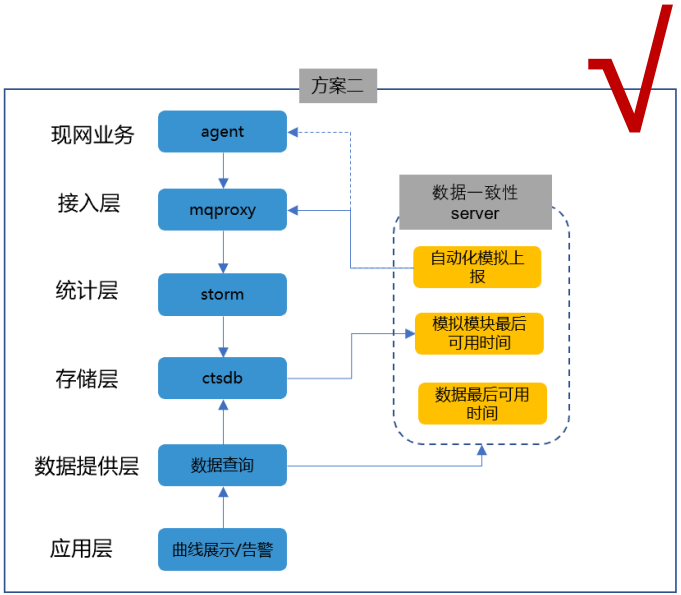
效果：发现很多次纬度值上报异常情况；发现超大纬度值笛卡尔积情况很多次；

故障处理：数据时效性与稳定性平衡方案

数据时效性和数据稳定性的矛盾？



VS

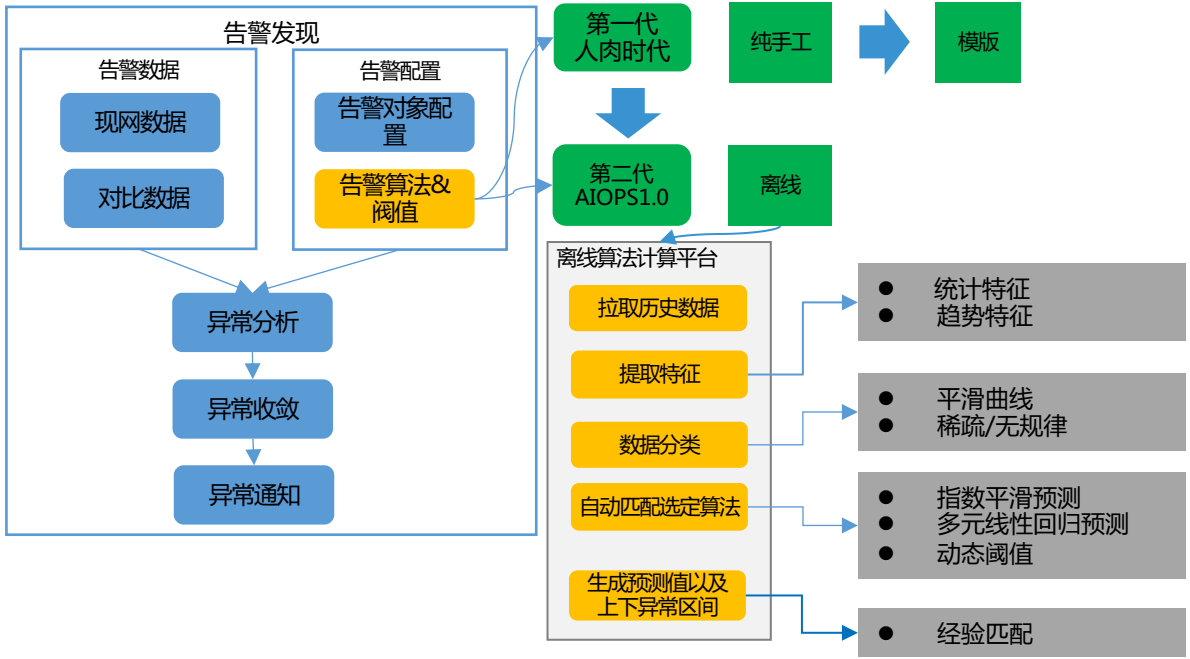


优势：	准确性高
劣势：	影响时效性 改造成本高，修改原链路逻辑

优势：	旁路系统，对原链路无侵入 实现简单 不影响时效性
劣势：	准确定一般(以偏概全)

故障处理：告警发现--AIOPS1.0

告警怎么配置都配置不准，要么漏、要么误告太多？



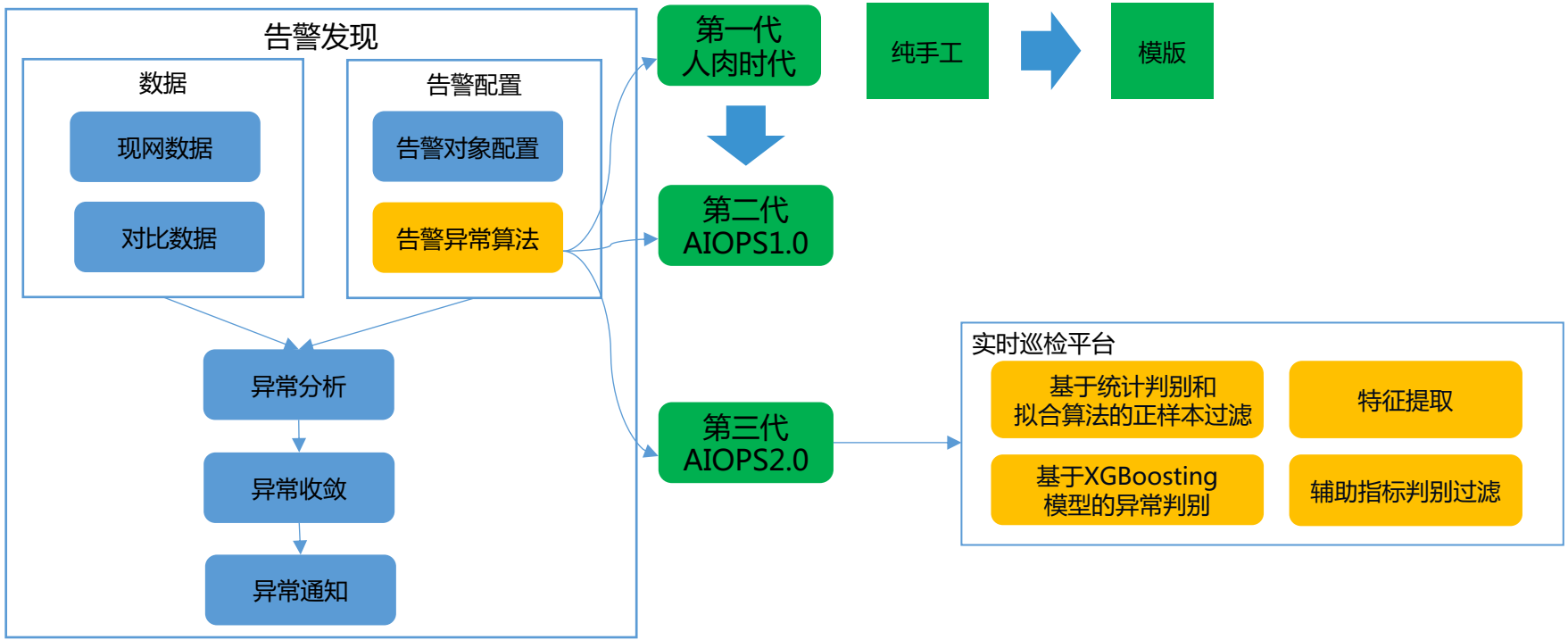
人工配置告警对象

+算法自动匹配

+人工经验阈值敏感度

+人工经验时间的敏感度

故障处理：故障发现--AIOPS2.0

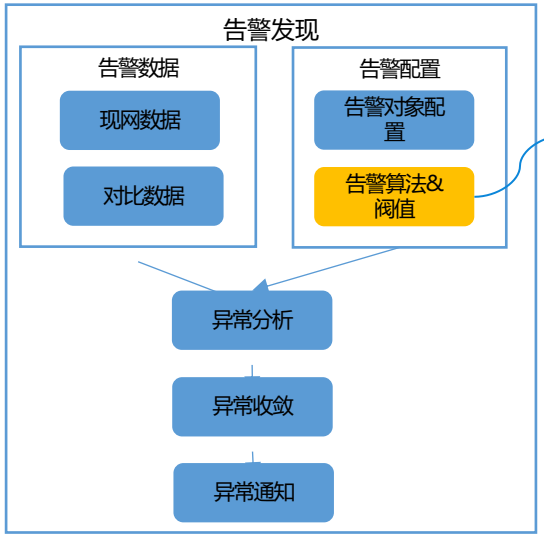


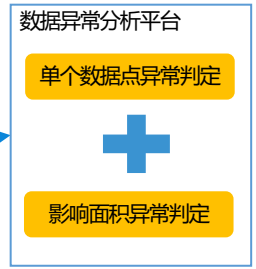
故障处理：故障发现--面积算法

1、这么明显的异常，为啥要连续好几次才告警，耽误了故障处理时间？

2、微涨持续了这么久，怎么就不能发现呢？

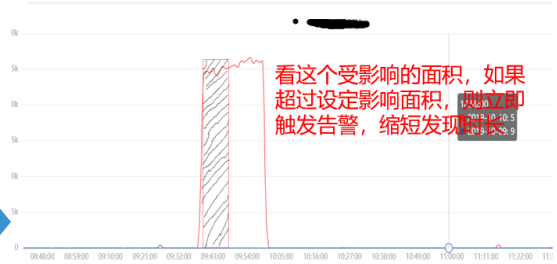
3、小波动一次，就不要告警出来了，晚上没法睡觉？





配置异常面积自动计算

- 运营事故管理规范折扣
- 故障处理时间SLA折扣



面积算法优势：

- 影响大的故障尽早发现
- 影响小的故障用时间换准确性
- 告警准确性和时效性矛盾解决
- 告警漏报性和准确性矛盾解决

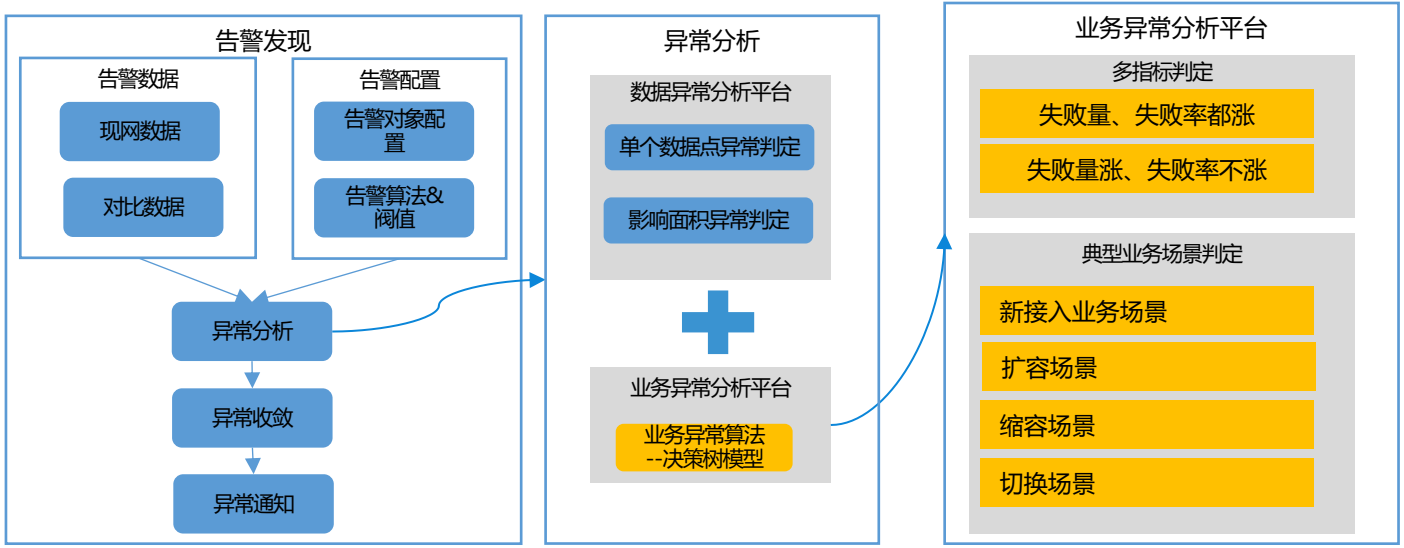
区间范围放小->降低漏报可能性
时间范围放大->提升告警准确性
面积算法->提升大故障告警时效性

终极效果：不在需要人为任何经验的配置告警，100%的自动化配置

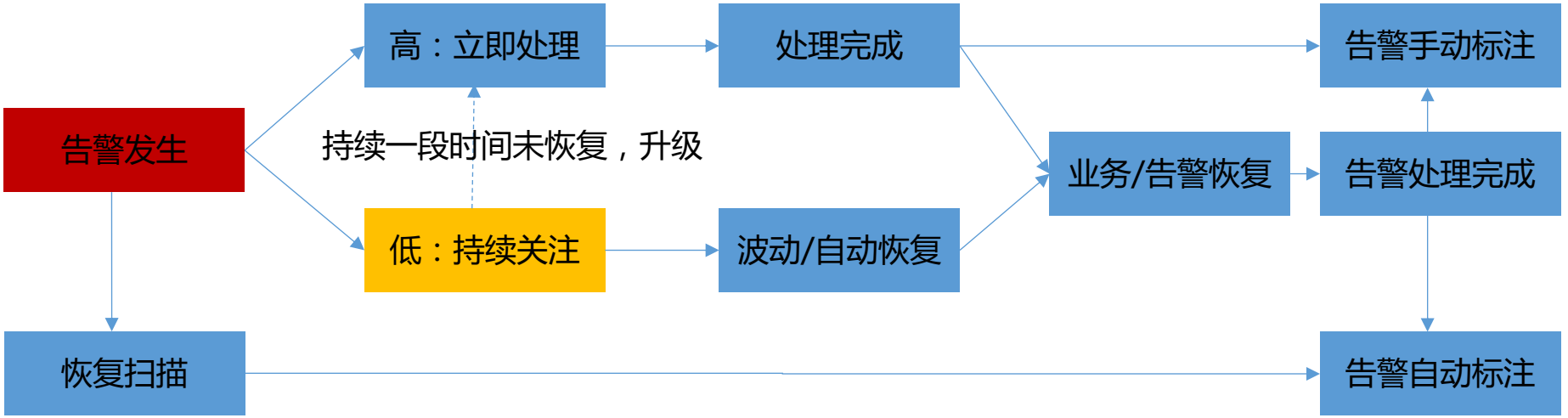
故障处理：故障发现--业务场景异常算法

1、下线了个set，怎么一直在告警？

2、今天业务搞个活动，怎么一天都在告警？



故障处理：故障发现--告警跟踪

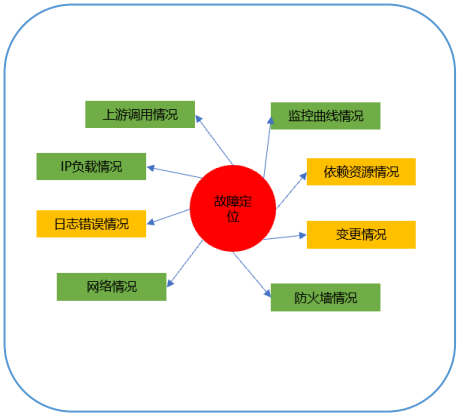


终极效果：让每一条告警都有跟踪处理和标注

故障处理：故障定位整体思路

故障定位之痛

调用多　链路长　部署复杂　各个运维系统割裂



双向分析法

一切以快速止损为目标

向上影响评估分析

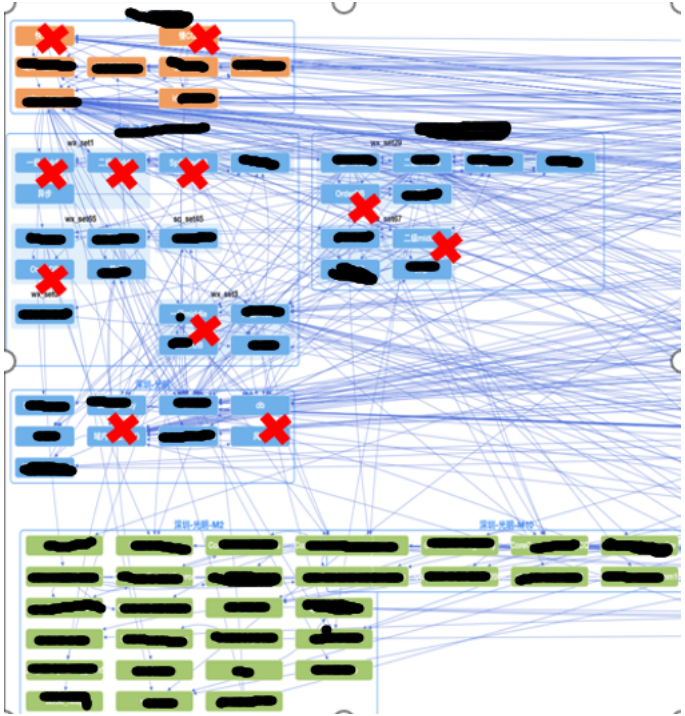
简要分析对用户侧的影响，以便上升决策，是否开大招

向下故障初因分析

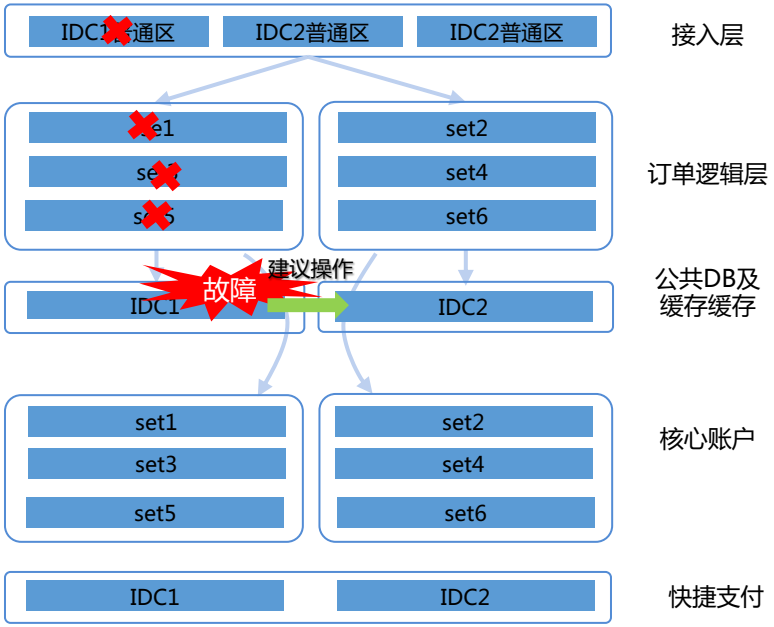
初步判断故障点，不要分析故障的根本原因，判断可恢复故障操作。

故障处理：故障定位--化繁为简

故障时，系统多处指标异常，到底那是纬度原因呢？

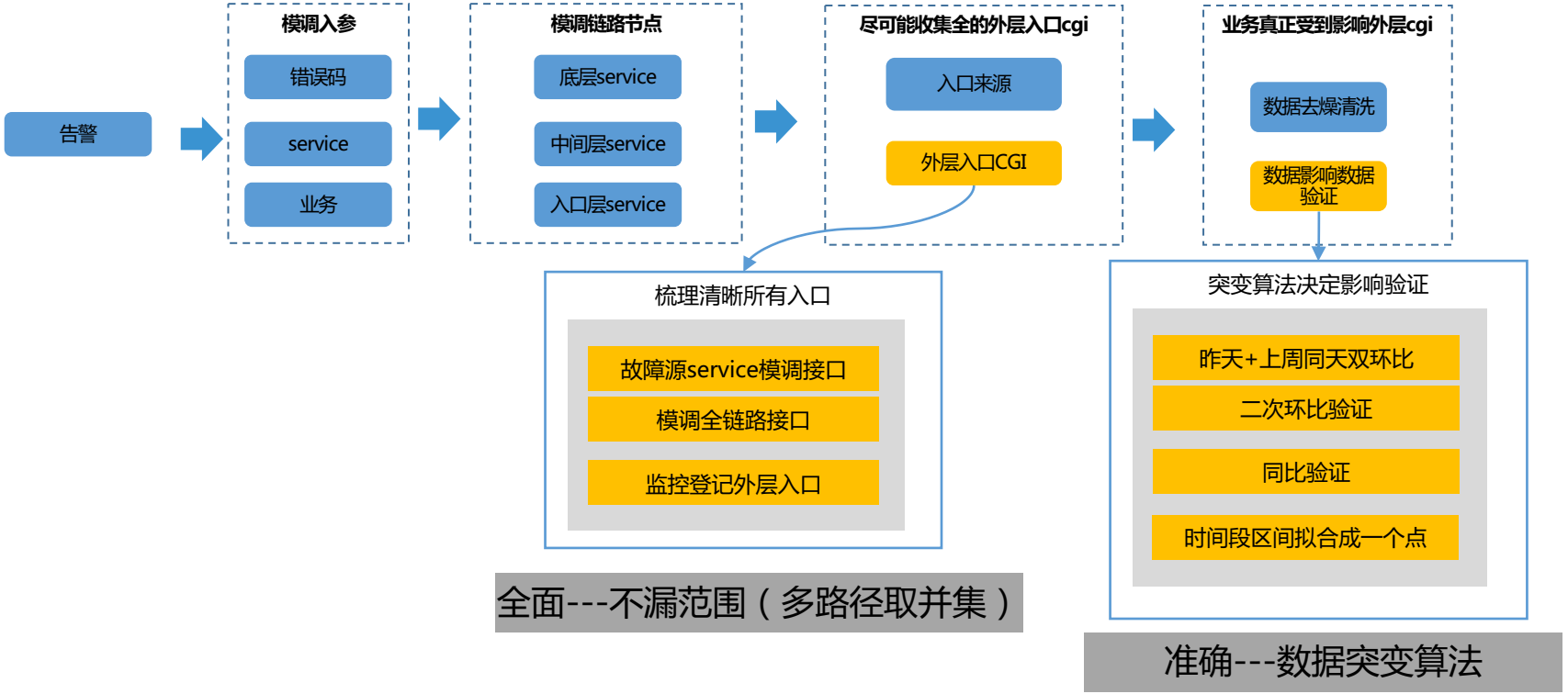


化繁为简

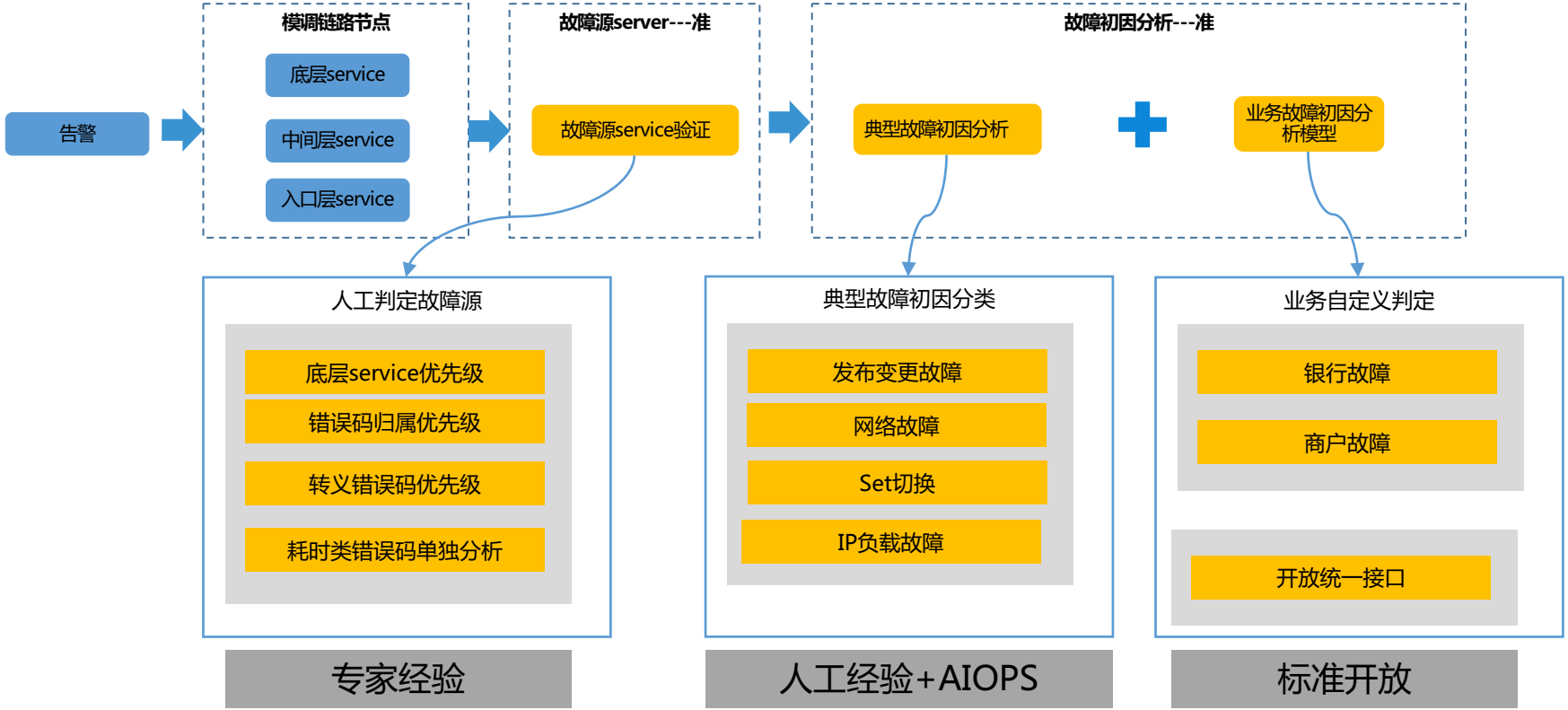


只找出故障点的集群，不找根本原因。

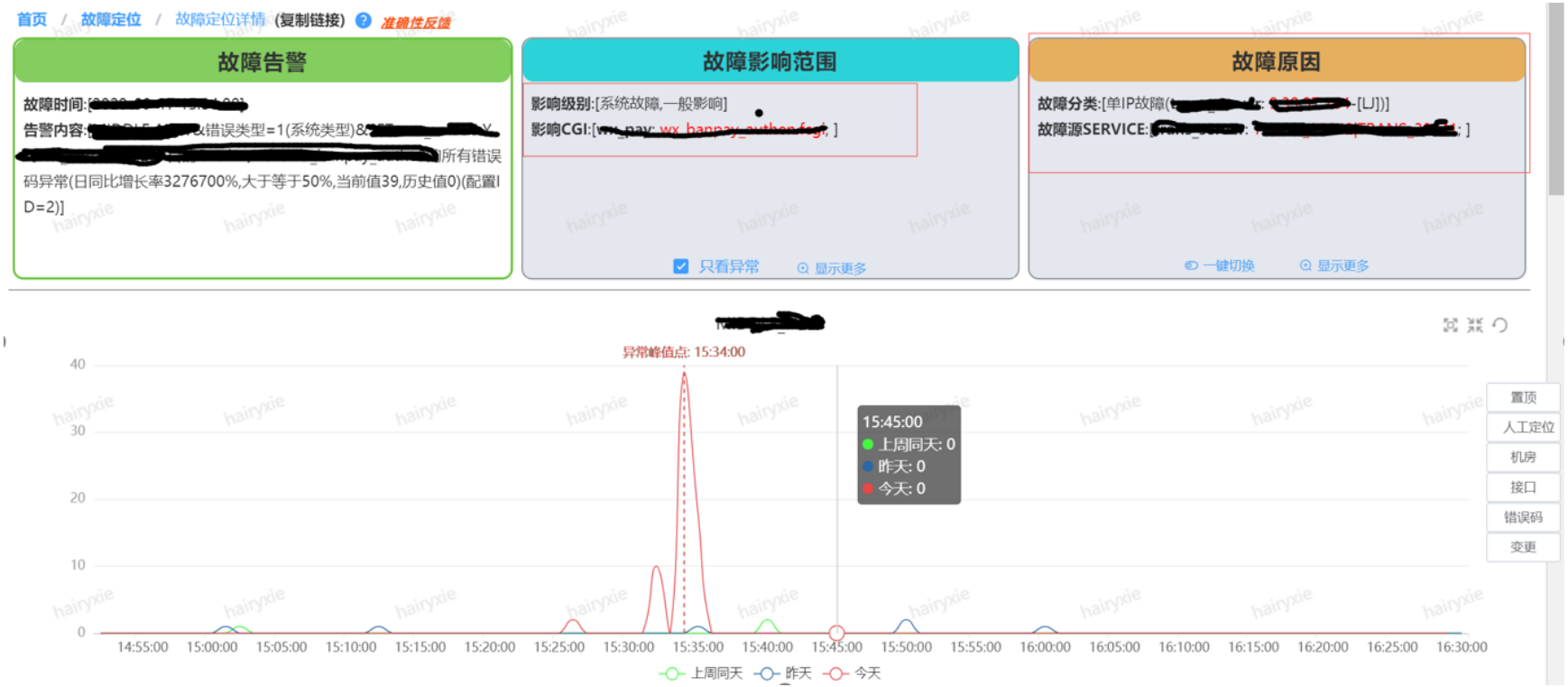
故障处理：故障定位-向上影响评估准确



故障处理：故障定位-向下初因分析准确

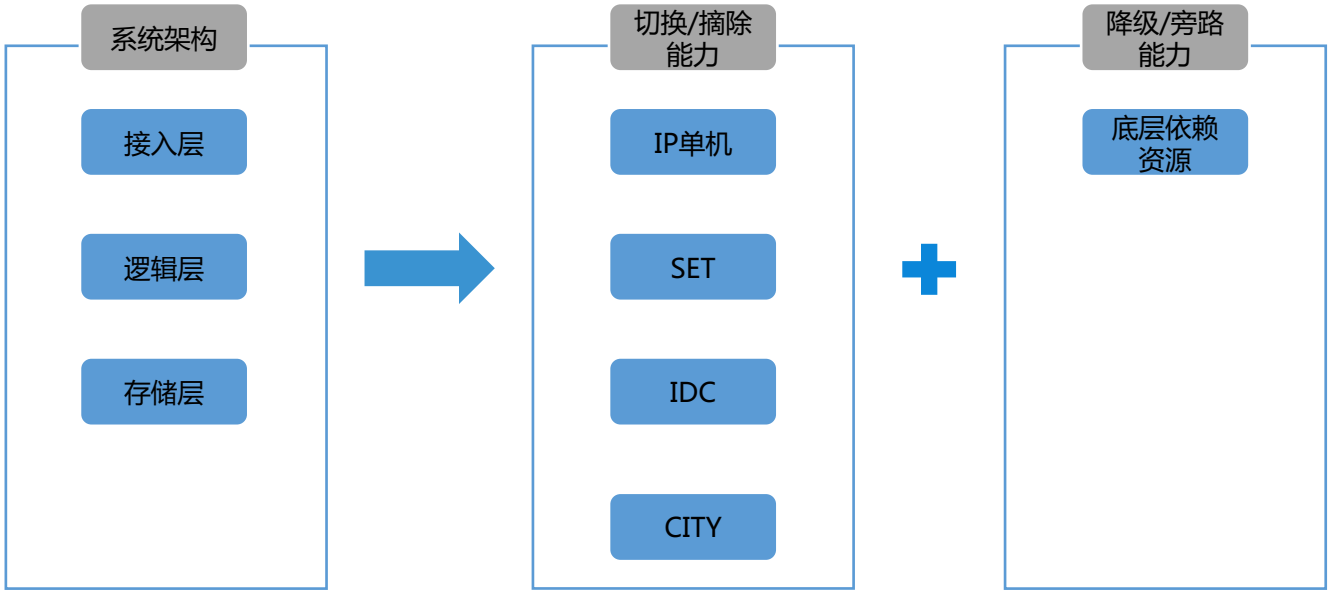


故障处理：故障定位--举例说明



故障处理：故障处理工具建设--容灾白皮书

如何建设故障处理能力的完备性？



故障处理：故障处理有效性保障--故障演习

如何确保，关键时刻切换工具是有效的？

实践是检验整理的唯一标准，通过演习保障自动切换和一键强制切换的有效性

单机

异常条件
① 网络异常
② 进程异常
③ cpu负载高

频率：每天
自动化发起一轮

单机房

异常条件：
① 全断网：网络完全中断
② 部分断网：单台交换机
③ 间歇性断网：网络抖动

频率：每月
人工发起

单城市

异常条件：
① 全断网：网络完全中断

频率：每半年
人工发起

平时：跨城单号段
每周1次自动演习

开关降级类

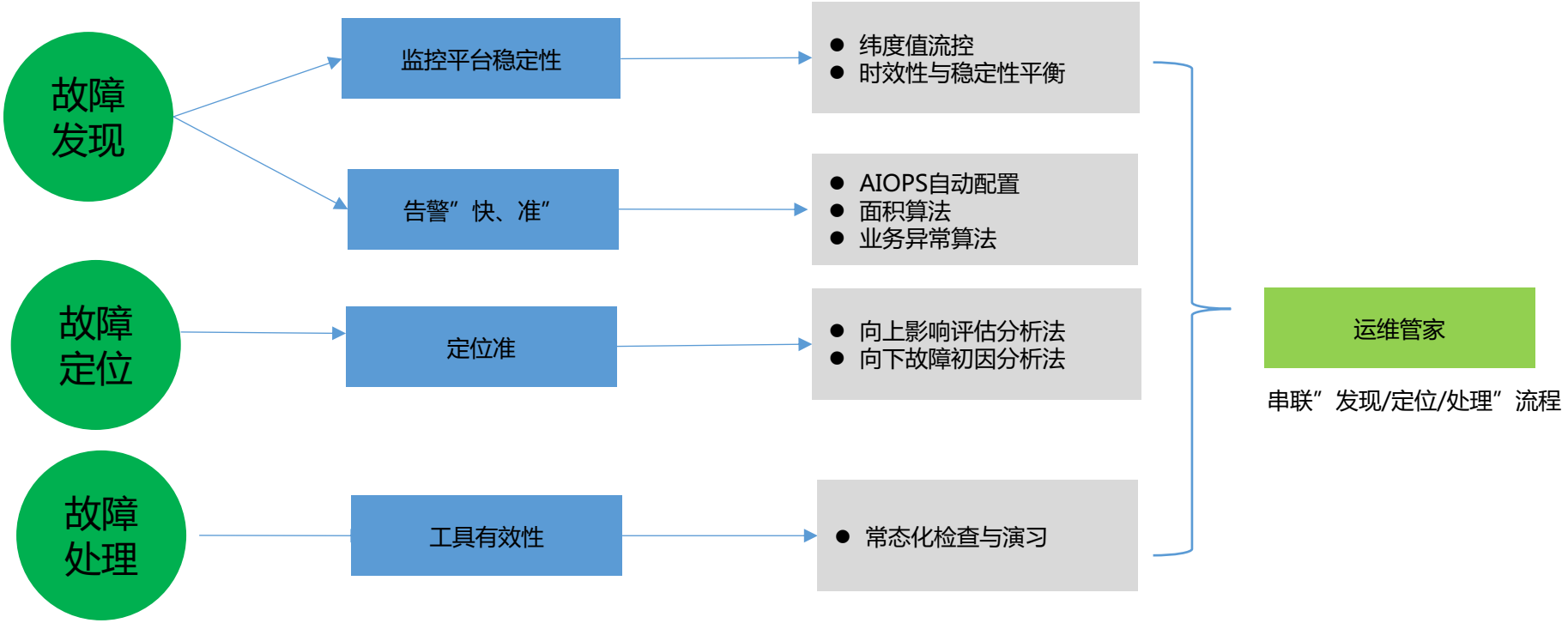
异常条件：
① 人为设置开关

频率：双周
一个人完成一个业务全部场景

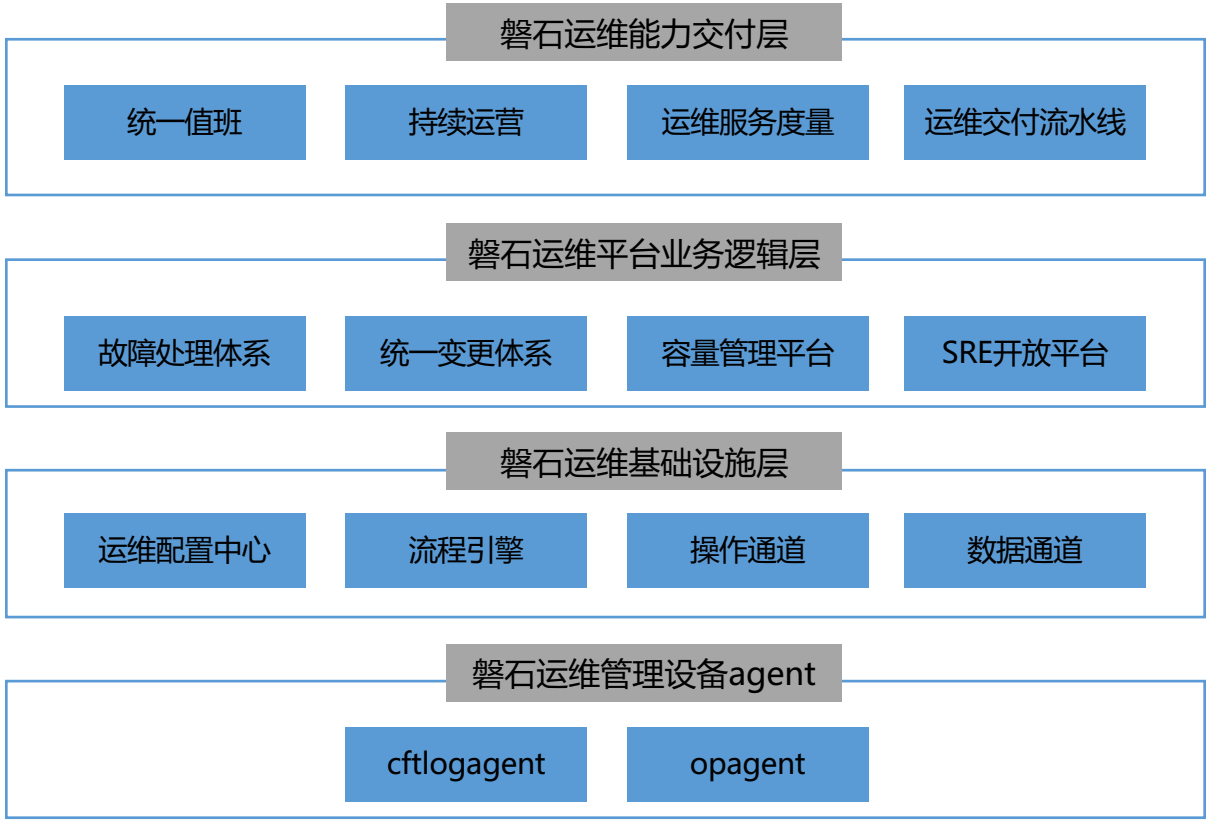
故障处理：故障处理提效--运维管家

终极效果：故障处理路径唯一化，7*24小时值班人员可以处理故障。

故障处理小结


故障发现
监控平台稳定性
纬度值流控
时效性与稳定性平衡
告警”快、准”
AIOPS自动配置
面积算法
业务异常算法
故障定位
定位准
向上影响评估分析法
向下故障初因分析法
故障处理
工具有效性
常态化检查与演习
运维管家
串联”发现/定位/处理”流程

运维平台未来思考



接入即服务：

整合“效能，组件，运维”

开放能力：

标准化+开放个性化的能力

智能运维：

故障自愈

自动发布

自动扩缩容