B站统一监控系统的设计,演进与实践分享

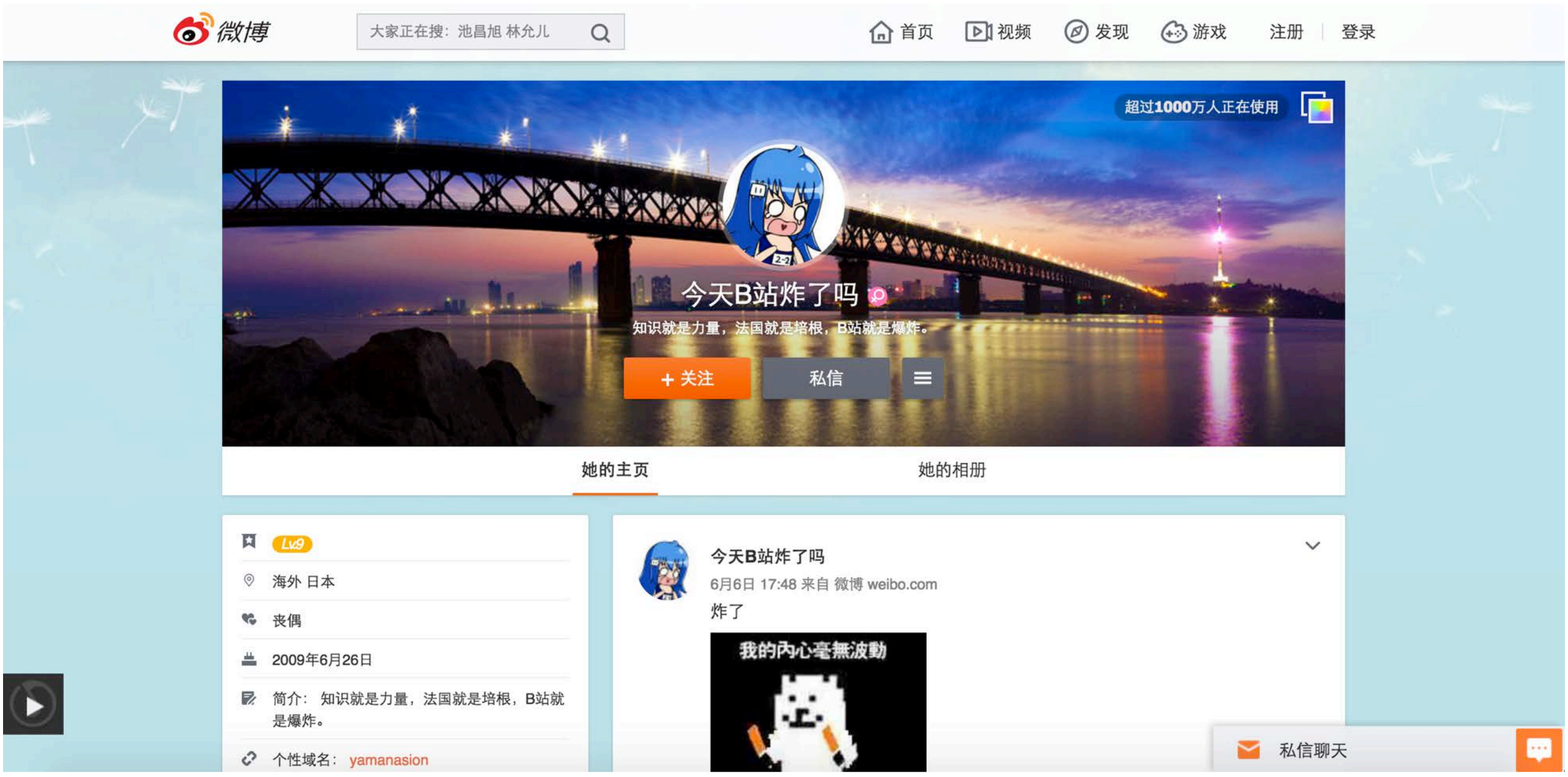
梁晓聪 devops
@lxcong

About Me

- 梁晓聪
- 2015年加入B站
- devops
- 热爱新技术,热爱开源
- 小宅男

故事的开始

B站炸了.舆情监控(括弧笑脸)



我们的挑战

当前情况:

- 技术栈多
- 产品模块复杂
- 业务爆发式增长
- 运维要求高

监控问题爆发:

- 覆盖率低
- 误报，漏报多
- 告警风暴

重新定义的监控系统

- 完整的监控体系
- 科学的告警策略
- 统一的告警中心

完整的监控体系

用户端监控

客户端质量

- 用户端网络质量
- 劫持情况
- 崩溃&卡顿
- 返回码
- 响应时间
- 错误率

播放质量

- 点播/直播
- 播放卡顿
- 平均首帧
- 播放失败率
- 弹幕加载
- cdn质量

服务端监控

业务层

- qps/tps
- 耗时分布
- 饱和度
- 吞吐量
- 依赖响应
- 缓存命中率
- 调用链
- SLA
- 日志

应用层

- cache资源
- db资源
- mq资源
- lb资源
- es资源
- 分布式文件
- 进程监控

基础层

- 虚拟机
- 物理设备
- 容器

- 专线质量
- 机房出口质量
- 交换设备

- http
- tcp
- ping

如何推进?

分析监控场景对应监控手段

场景	类型	手段
服务端监控	metric类型	时间序列数据
	日志类型	日志处理流
	自定义类型	自研
用户端监控	客户端	apm
	播放器	自研

如何推进?

分析监控场景对应监控手段

场景	类型	手段
服务端监控	metric类型	时间序列数据
	日志类型	日志处理流
	自定义类型	自研
用户端监控	客户端	apm
	播放器	自研

metric方案选型

metric数据特征

- 能覆盖大部分监控场景
- 固定几种数据类型
  - ✦ Counter
  - ✦ Gauge
  - ✦ 等..
- 时序数据
  - ✦ 具有统计特性
  - ✦ 具有规律性

选型原则

- 基于开源方案，二次开发
- 具备现代时间序列数据库的特性
- 活跃项目,具有成熟的生态环境

结论

- **prometheus**
- 支持任意维度label
- cncf基金会

metric

现状:

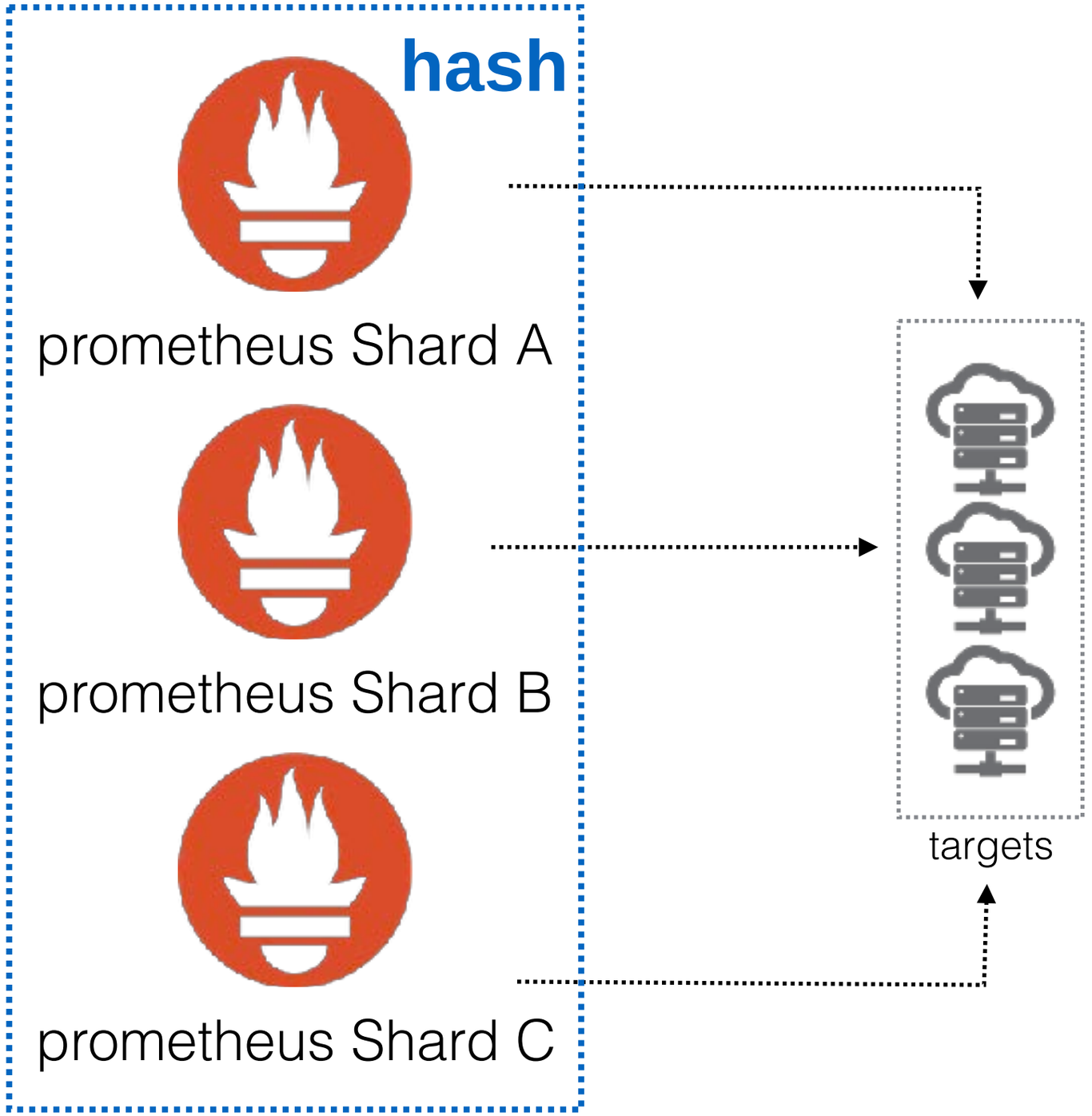
- 40w+/s的指标采集
- 10k+ 监控目标
- 10+ prometheus节点

问题:

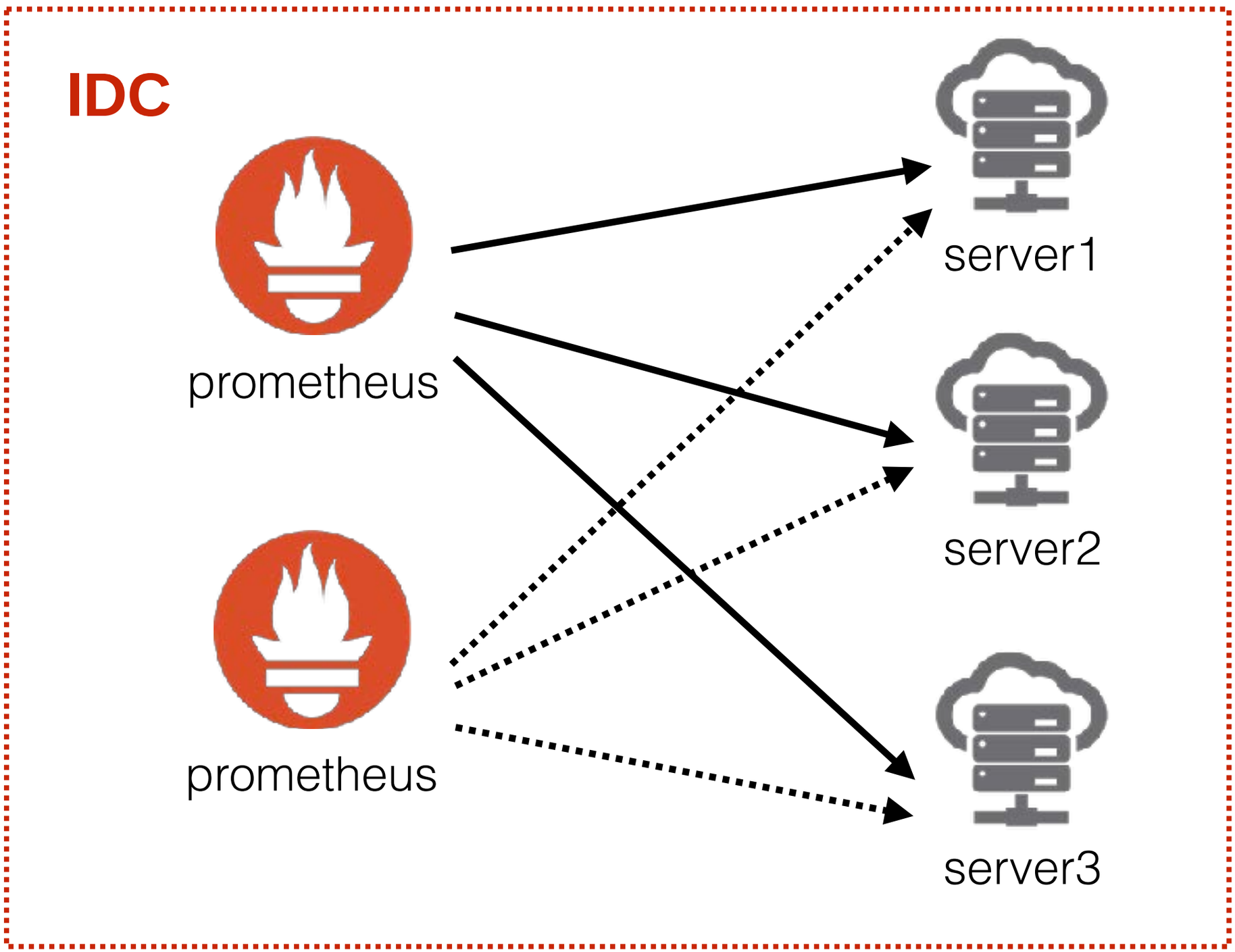
- 性能
- 高可用 ？
- 分布式
- 使用成本

性能问题

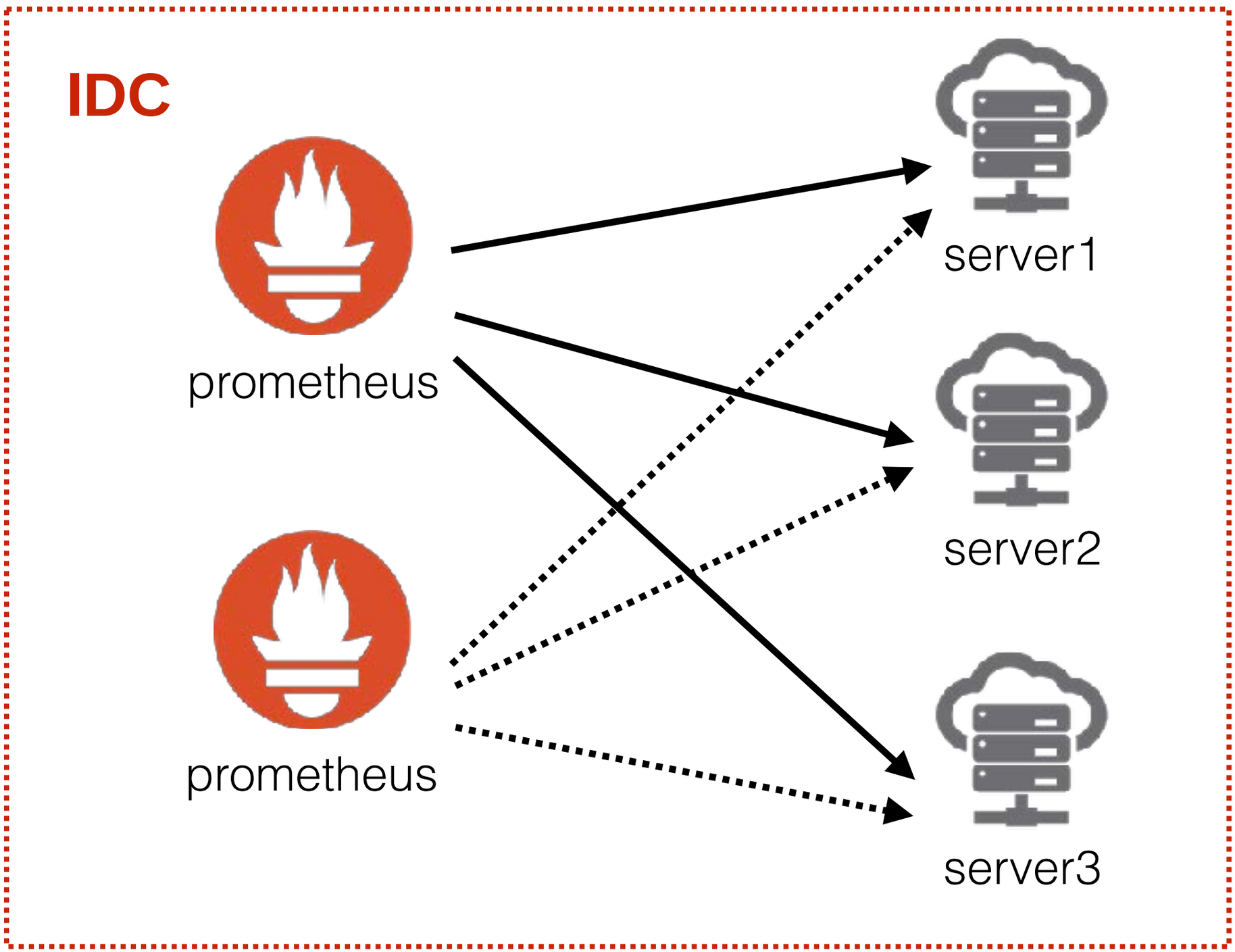
- 本地ssd
- horizontal sharding (实验性质使用)
- prometheus 2.0 (tsdb)



HA



HA

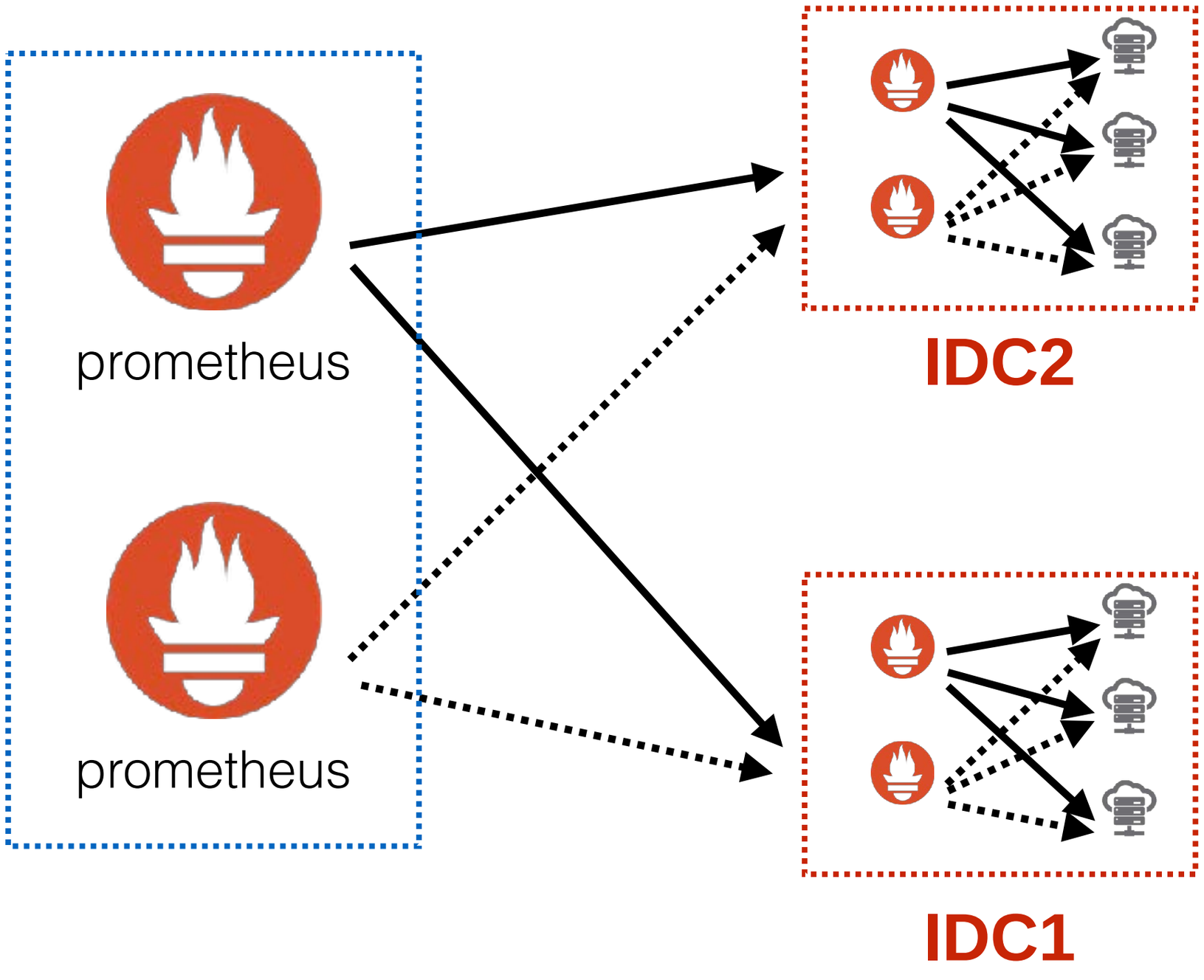


Federation

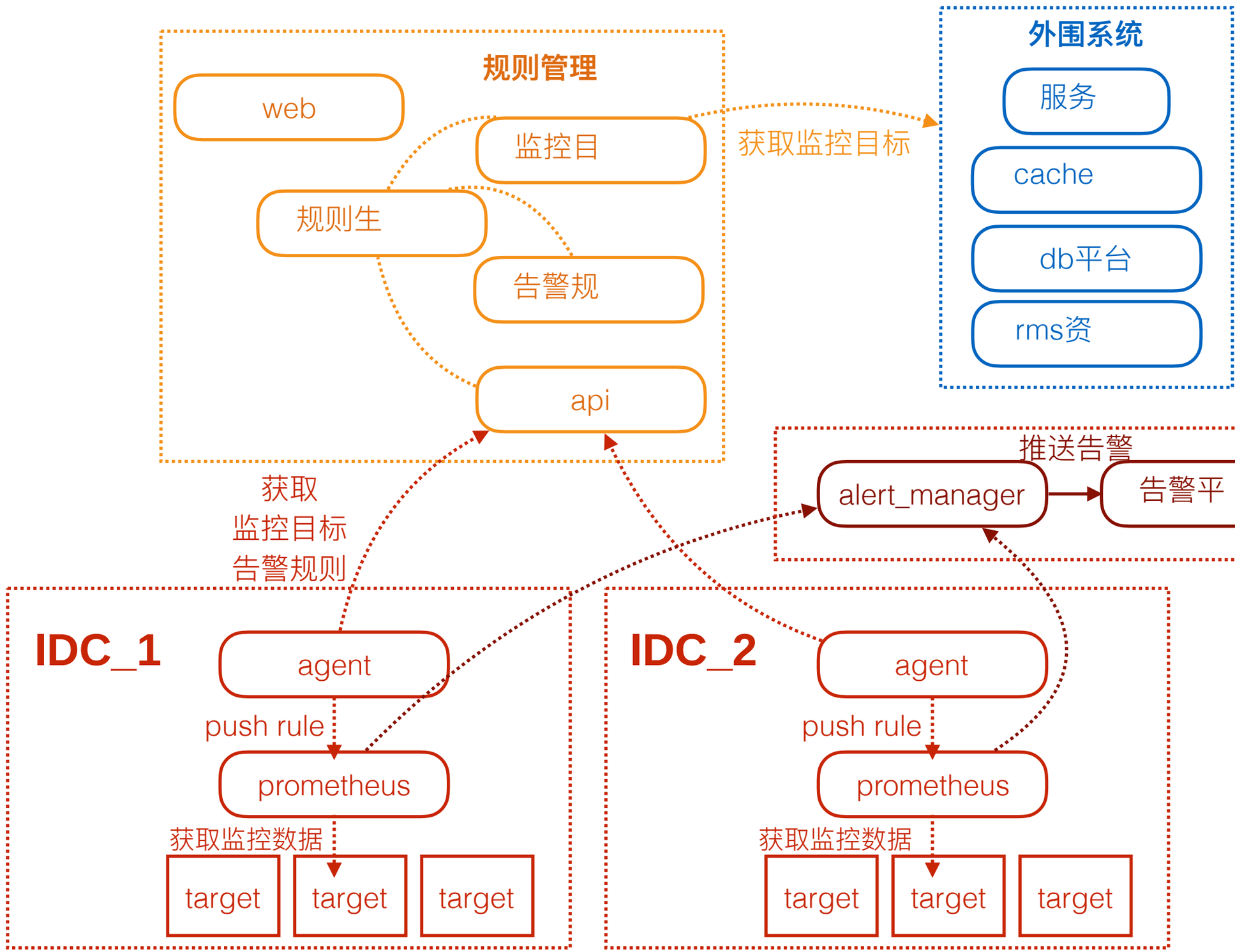
建议

filter数据

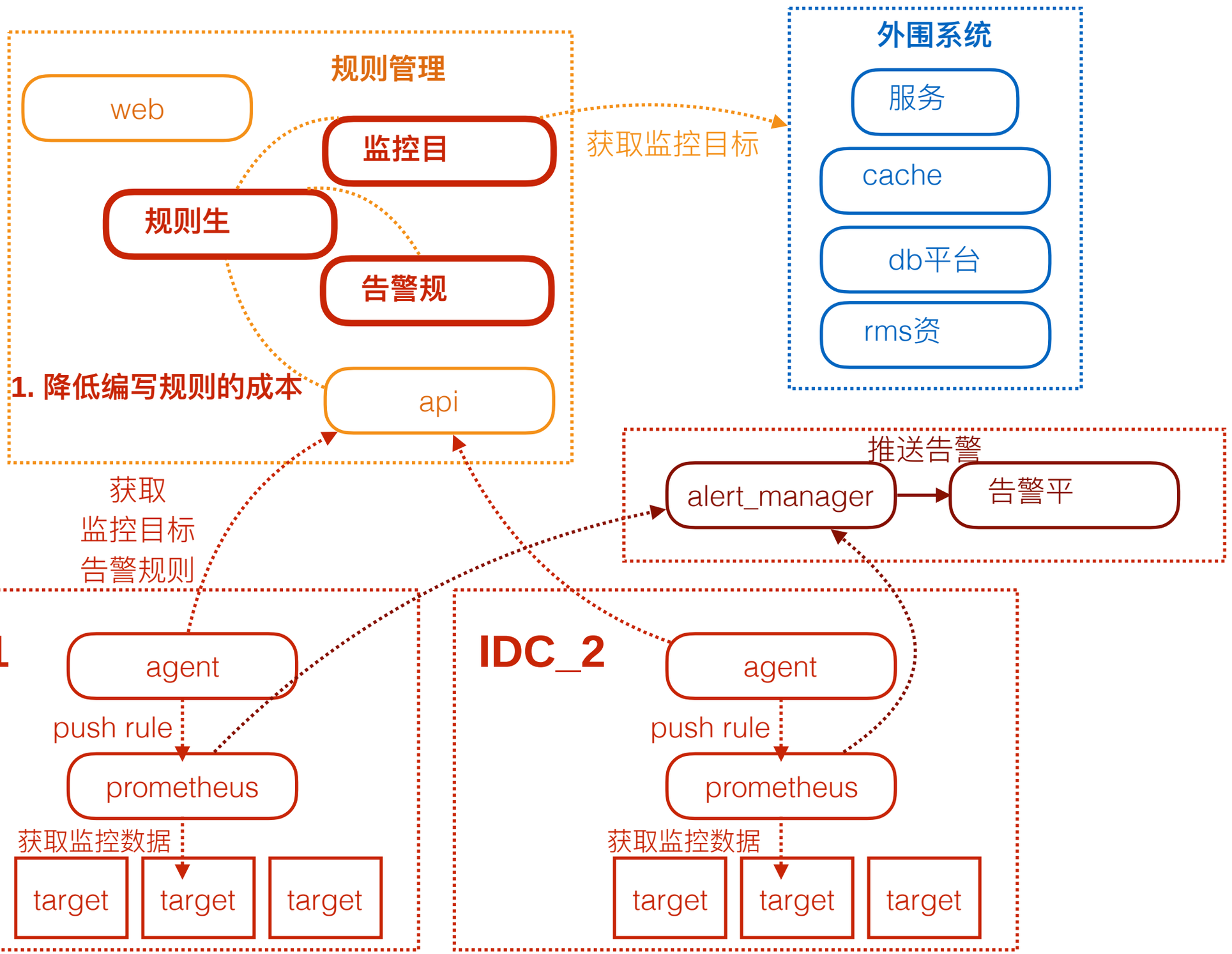
精度降低



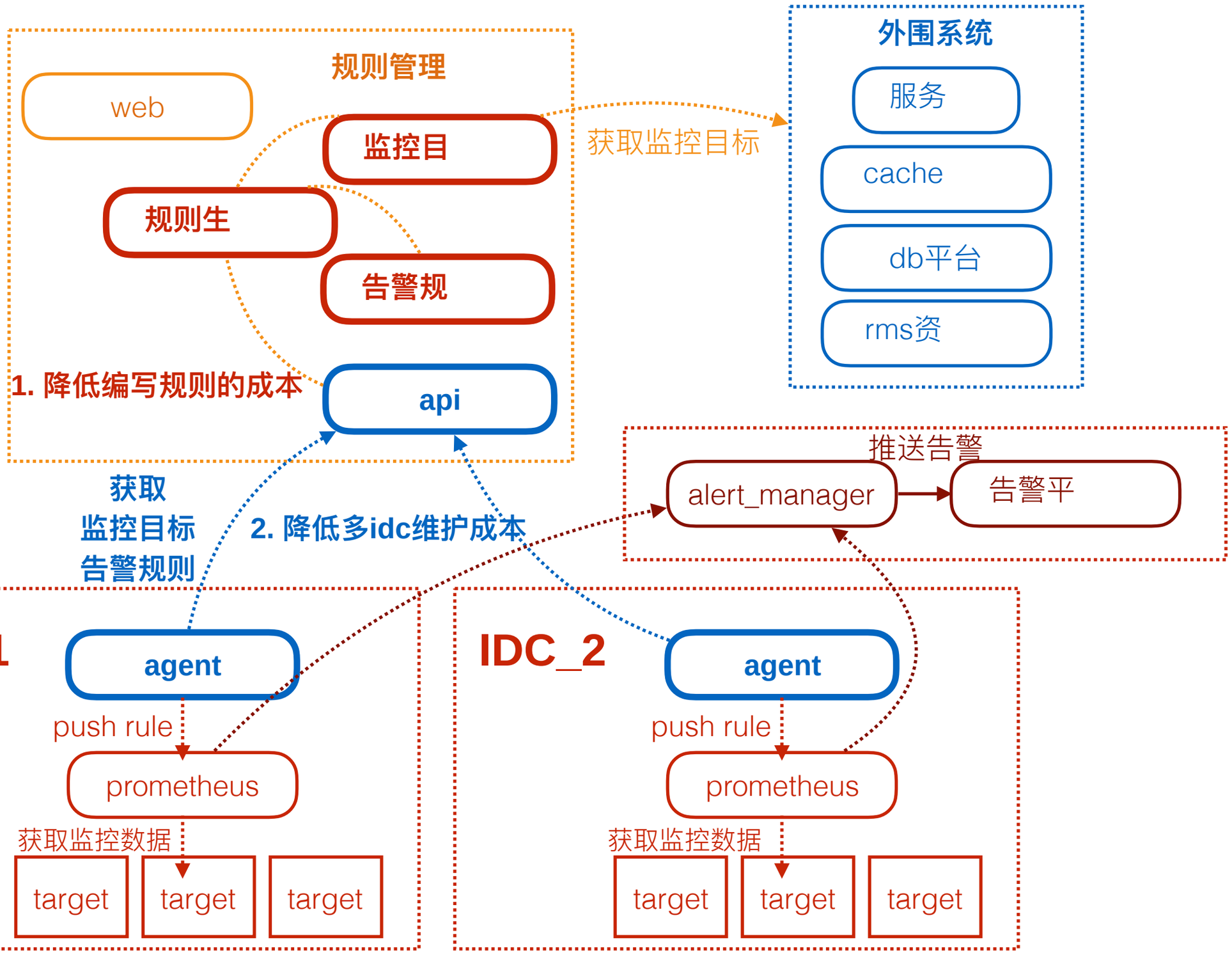
降低使用成本


规则管理
web
监控目
规则生
告警规
api
获取监控目标
外围系统
服务
cache
db平台
rms资
推送告警
alert_manager
告警平
获取
监控目标
告警规则
IDC_1
agent
push rule
prometheus
获取监控数据
target
target
target
IDC_2
agent
push rule
prometheus
获取监控数据
target
target
target

降低使用成本


规则管理
web
监控目
规则生
告警规
api
1. 降低编写规则的成本
获取监控目标
外围系统
服务
cache
db平台
rms资
推送告警
alert_manager
告警平
获取
监控目标
告警规则
IDC_1
agent
push rule
prometheus
获取监控数据
target
target
target
IDC_2
agent
push rule
prometheus
获取监控数据
target
target
target

降低使用成本


规则管理
web
监控目
规则生
告警规
1. 降低编写规则的成本
api
获取监控目标
外围系统
服务
cache
db平台
rms资
推送告警
alert_manager
告警平
获取
监控目标
告警规则
2. 降低多idc维护成本
IDC_1
agent
push rule
prometheus
获取监控数据
target
target
target
IDC_2
agent
push rule
prometheus
获取监控数据
target
target
target

规则管理页面

Bili Monitor

切换应用

lxcong

1 关联资源

* 产品：

* 资源类型：

应用分组：

2 设置报警规则

* 规则描述：cpu_load | 5分钟 | 平均值avg | > | 30 | % | AND | 拓展 | 删除

* 规则描述： | >= | 阈值 | % | 拓展 | 删除

+添加报警规则

持续时间：1 (分钟)

生效时间：00:00 至 00:00

3 通知方式

通知对象：554 联系人通知组

Search

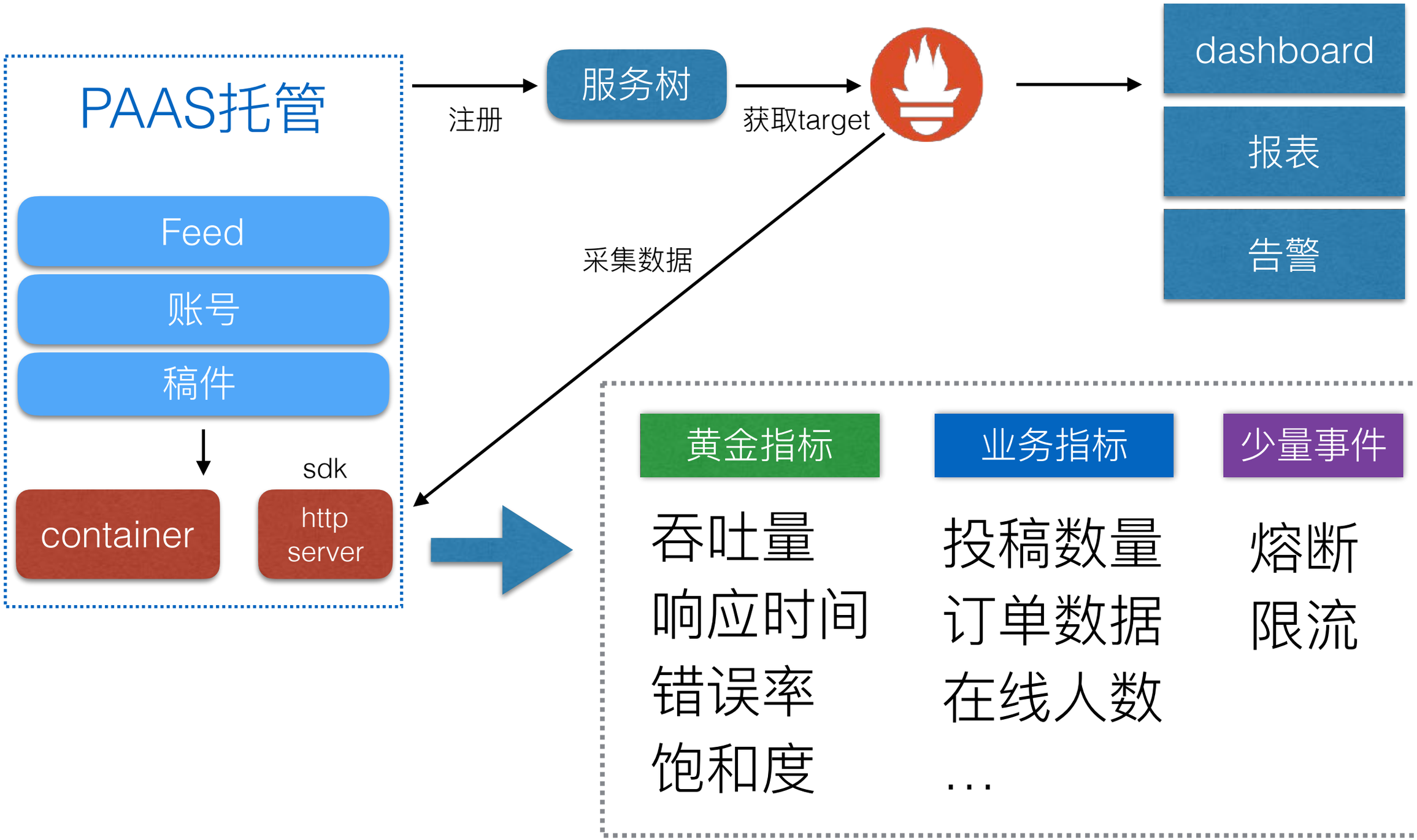
- dev_alarm_group
- /game/gfate/yc/运维人员
- /game/fate/ts/运维人员
- /game/fate/test/运维人员
- /game/fate/bc/运维人员
- /game/ags/bs/运维人员
- /game/ags/cb2-all/运维人员

已选组

Search

- /game/fate/dev/运维人员

例子 - 业务监控

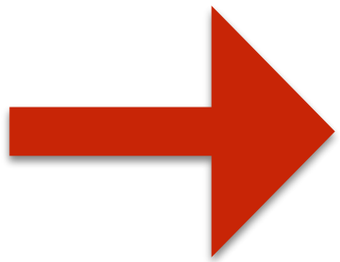

PAAS托管
Feed
账号
稿件
container
sdk
http server
注册
服务树
获取target
采集数据
dashboard
报表
告警
黄金指标
吞吐量
响应时间
错误率
饱和度
业务指标
投稿数量
订单数据
在线人数
…
少量事件
熔断
限流

统一的告警中心

解决什么问题?

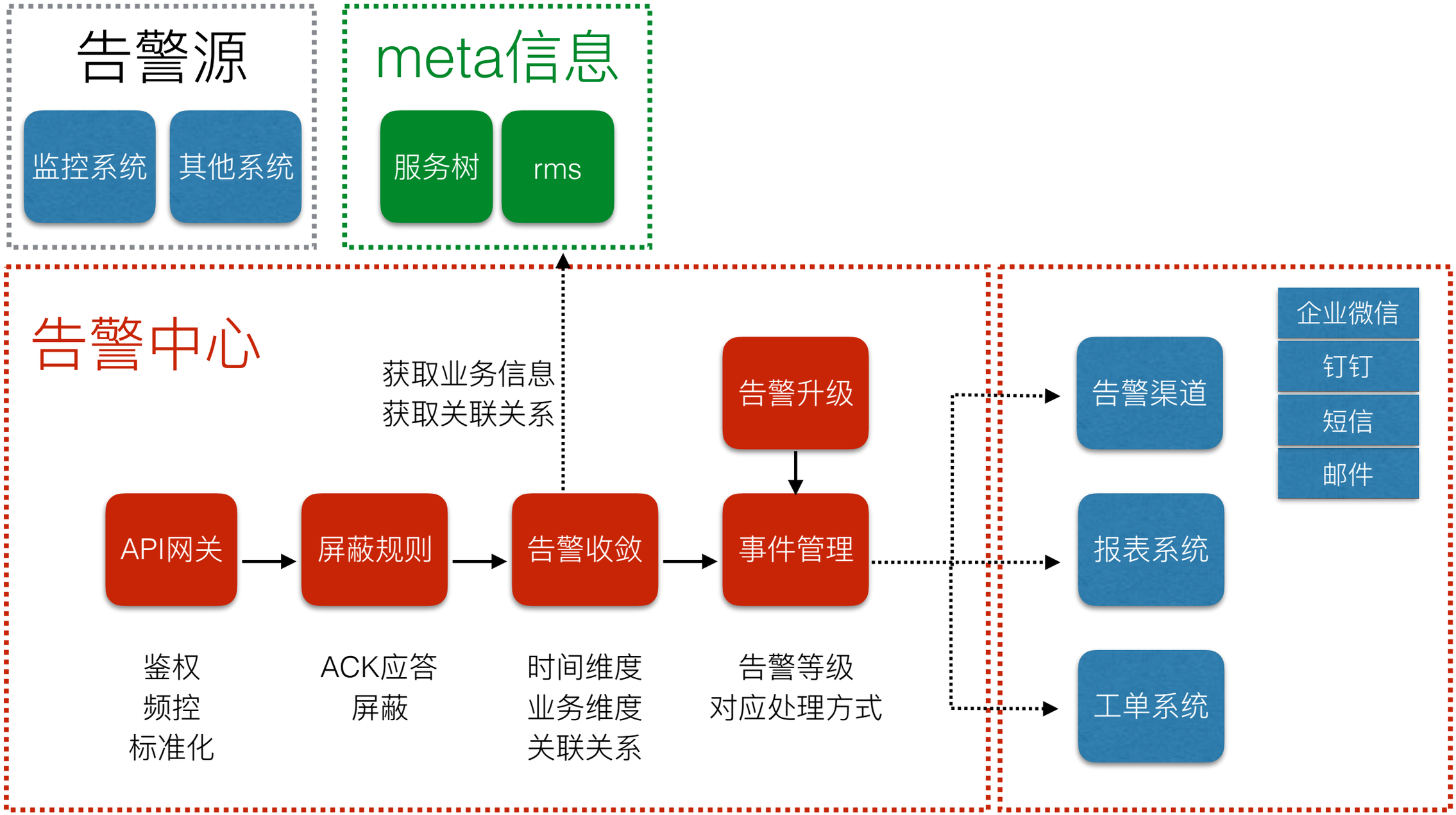
问题

- 告警源头多
- 告警风暴, 大量重复告警
- 发送告警渠道多
- 重要告警没有及时到达
- 优化告警没有数据依据

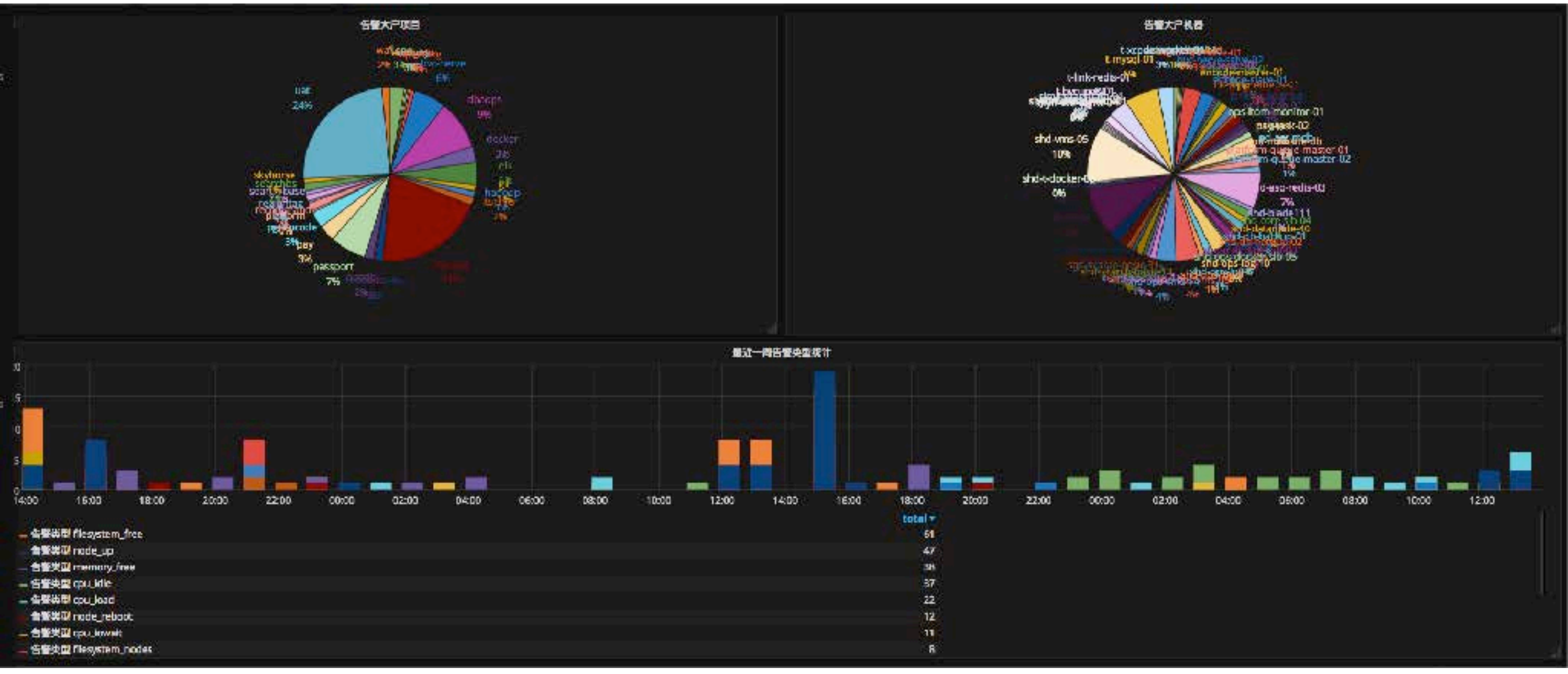


核心功能

- 告警标准化
- 告警收敛
- 告警渠道管理
- 告警升级
- 告警报表


告警源
监控系统
其他系统
meta信息
服务树
rms
告警中心
获取业务信息
获取关联关系
API网关
屏蔽规则
告警收敛
告警升级
事件管理
鉴权
频控
标准化
ACK应答
屏蔽
时间维度
业务维度
关联关系
告警等级
对应处理方式
告警渠道
报表系统
工单系统
企业微信
钉钉
短信
邮件

告警统计


告警大户项目
告警大户机器
最近一周告警类型统计
total
告警类型 filesystem_free 61
告警类型 node_up 47
告警类型 memory_free 38
告警类型 cpu_idle 37
告警类型 cpu_load 22
告警类型 node_reboot 12
告警类型 cpu_iowait 11
告警类型 filesystem_nodes 6

有意思的尝试


新版[告警]
waf [redacted] ip_conntrack表使用率 99.66%
[redacted]
下午2:00
新版[恢复]
livedm [redacted] 节点宕机(无法Ping通)
restart [redacted]
确认重启
Server Code: [redacted]
Server Name: [redacted]
请回复 你懂的暗号
[redacted]
提交任务成功,返回结果:Chassis Power Control: Reset

科学的告警策略

科学?

~~machine learning?~~

~~deep learning?~~

不要盲目的使用机器学习

先让告警有意义

可读的

- 时间
- 源头
- 规则
- 影响
- 状态

正确的

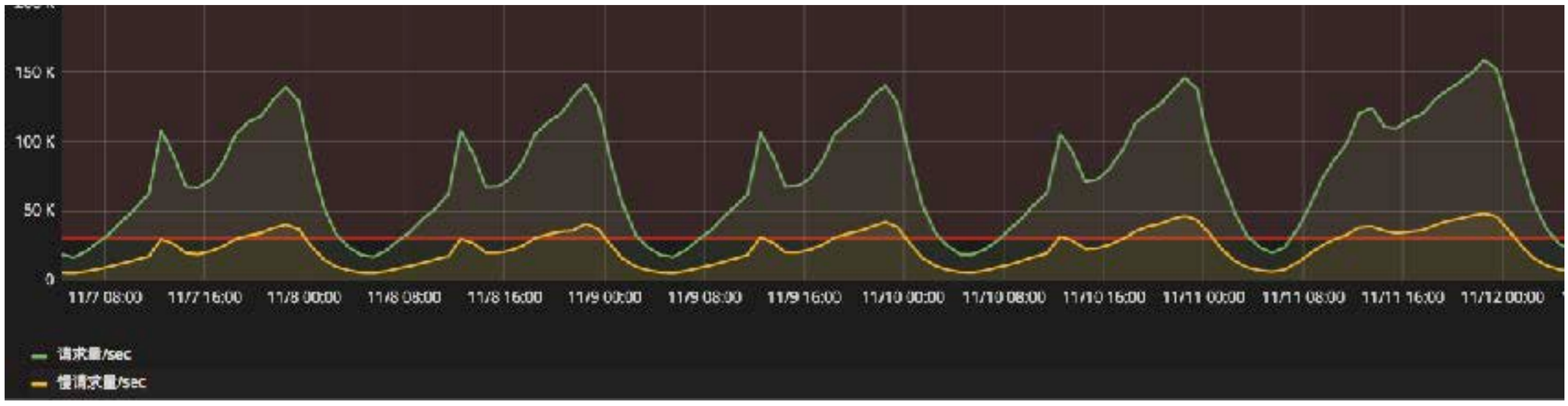
- 正确反映现实

有价值的

- 发现问题

案例1

告警规则: 业务A 慢请求量 > 10k/s **wrong**



固定阈值

告警阈值需要随着流量变化而调整

建议:

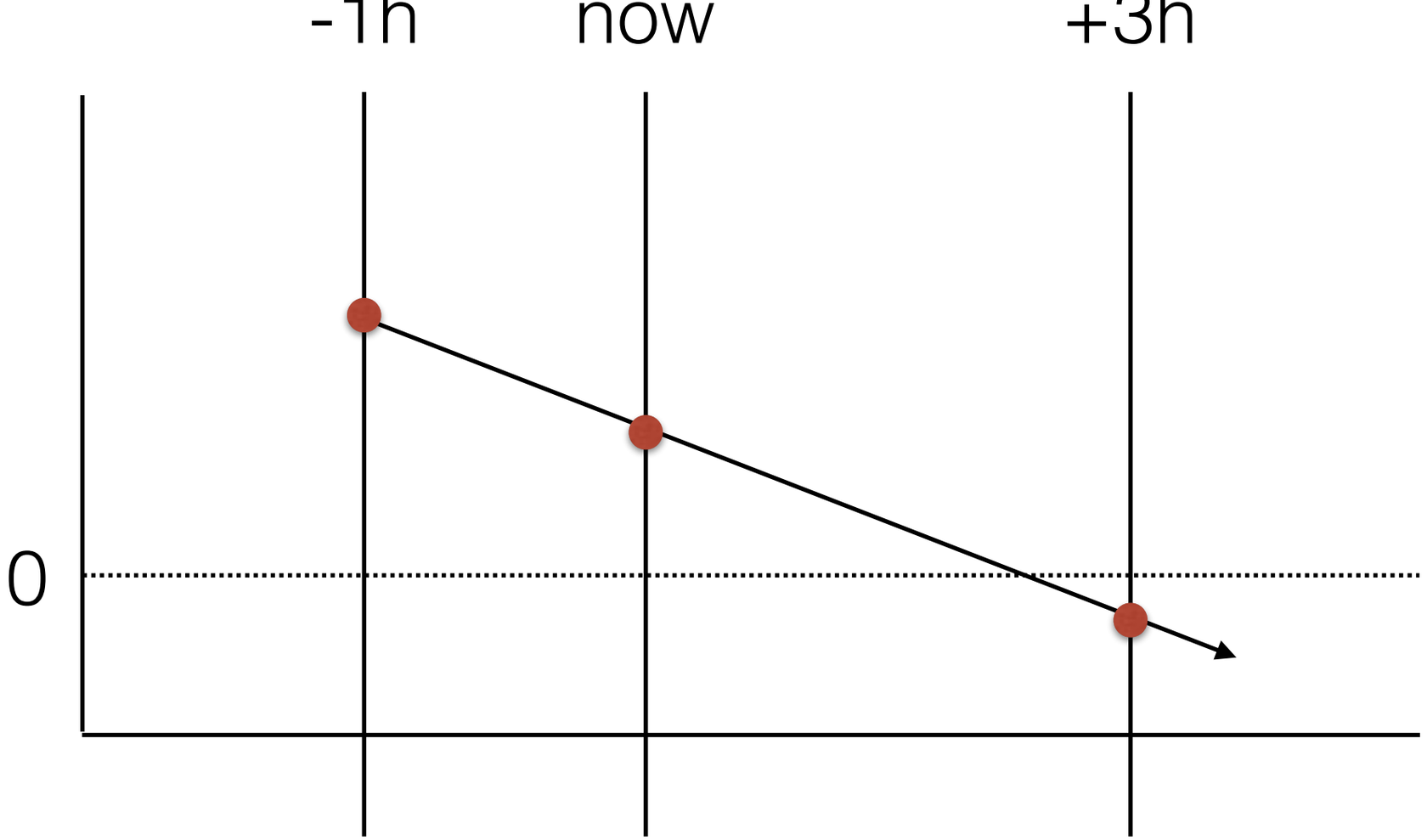
告警规则: 业务A 慢请求比例 > 80%

案例2

告警规则: 磁盘容量可用率 <10%

告警规则: 磁盘容量预计将于3小时后饱和

predict_linear(node_filesystem_free{}[1h], 3 * 3600) < 0

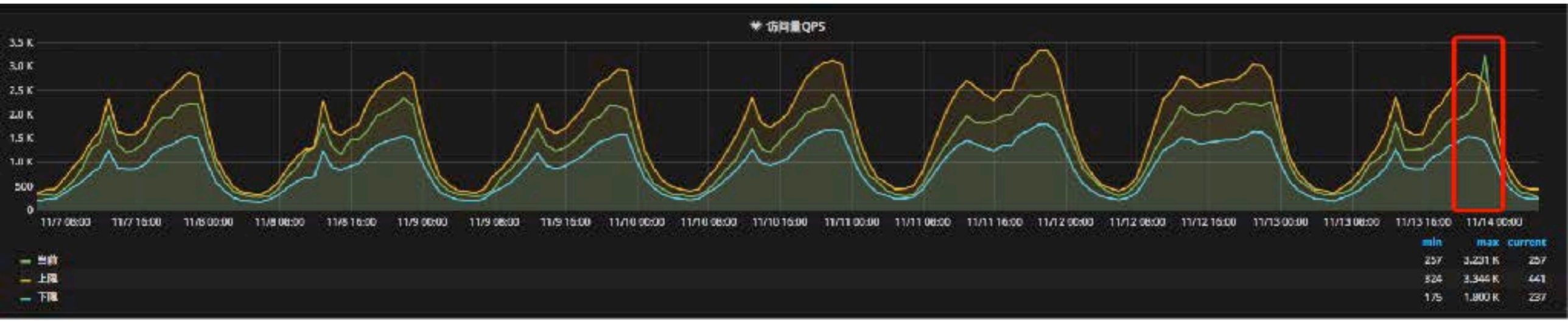


异常检测

异常流量

告警规则: 预测业务A请求量异常

abs(requests - requests:holt_winters_rate1h offset 7d) > 0.3 * requests:holt_winters_rate1h offset 7d



异常响应

```
(
    instance:latency_seconds:mean5m
  > on (job) group_left()
    (
        avg by (job)(instance:latency_seconds:mean5m)
      + on (job)
        2 * stddev by (job)(instance:latency_seconds:mean5m)
    )
)
> on (job) group_left()
  1.2 * avg by (job)(instance:latency_seconds:mean5m)

and on (job)
  avg by (job)(instance:latency_seconds_count:rate5m) > 1
```

todo

针对历史事件

- 异常事件关联关系挖掘
- 全联路模块调用分析
- 瓶颈分析

针对当前事件

- 异常检查(动态阈值)
- 异常定位(根因分析)
- 快速止损

针对未来事件

- 故障预测
- 容量预测
- 趋势预测

Thank You!

哔哩哔哩 - (゜-゜)つロ 乾杯~ - bilibili