



# 稳定性方法论：可灰度 & 可监控 & 可回滚

京东云开发者

2024-03-22

👁 3,749

🕒 阅读9分钟

智能总结 ⓘ

复制

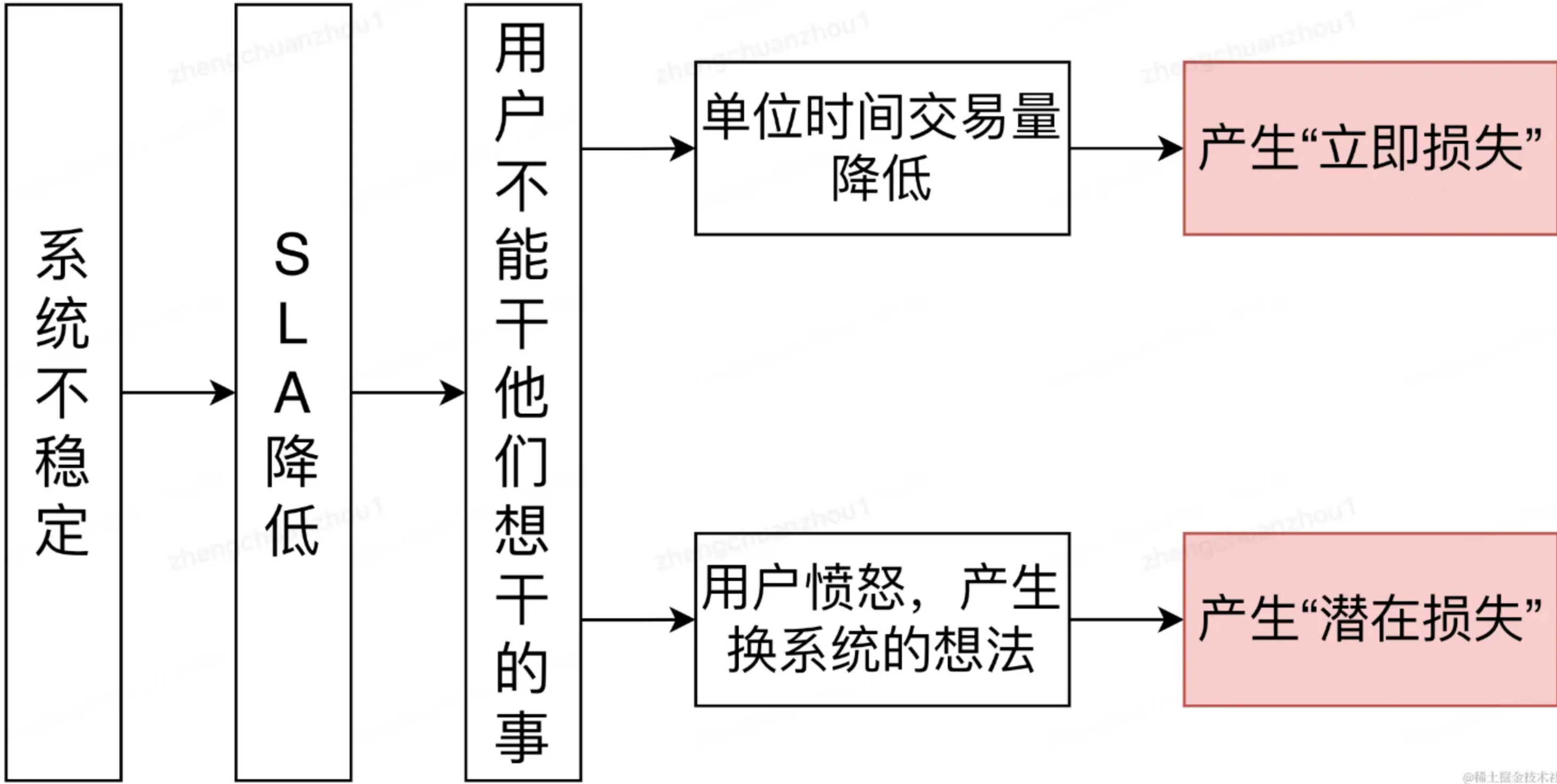
重新生成

这篇文章主要介绍了稳定性方法论“可灰度、可监控、可回滚”。可灰度包括机器灰度、AB 灰度、全链路灰度、沙箱灰度等方式及各自优劣势；可监控涵盖机器、链路、网络、业务等全方位监控，并列举了京东的相关监控系统及核心指标；可回滚是兜底，要从应用、数据库 DDL、数据等多个角度评估回滚相关事宜。

关联问题: [全链路灰度难在哪](#) [监控怎样平衡指标](#) [如何评估可回滚](#)

基于该文章内容继续向AI提问

业务系统核心目标是挣钱，系统稳定性建设核心是防止丢钱（丢钱逻辑如下图所示），站在公司的角度看，产品功能建设和系统稳定性是同等重要。



前段时间写了《[稳定性治理框架](#)》，该文章在稳定性建设的理论和实践基础上，抽象出稳定性治理的框架，希望建立一个稳定性治理的标准动作、最佳实践。但从读者的反馈上看，有过类似经验的同学深同感触，经验不足的同学没啥感觉，导致这个结果的原因，我反思了一下，认为：概念太粗，落地容易变形。于是，想写一篇文章，把稳定性最重要的东西写出来，于是有了这篇文章。

通常，导致线上事故的最大诱因是“变更”，“可灰度、可监控、可回滚”的方法论，目标就是降低变更引发的线上事故，其逻辑是：首先，通过灰度手段让变更可控，从0%的影响逐渐放量到100%影响；其次，通过监控手段观察变更是否准确，如果准确了，灰度放量才能继续；最后，如果出现不可控的情况，能够立刻回滚，立刻止损，避免长时间影响客户。

## 一、可灰度

可灰度是指线上变更能够支持灰度发布，百度百科对对灰度发布的定义是：灰度发布又名金丝雀发布，指在黑与白之间，能够平滑过渡的一种发布方式。在其上可以进行A/B testing，即让一部分用户继续用产品特性A，一部分用户开始用产品特性B，如果用户对B没有什么反对意见，那么逐步扩大范围，把所有用户都迁移到B上面来。

各大厂常用的灰度发布种类有：机器灰度、AB灰度、链路灰度、沙箱灰度等，下面展开讲解：

### •机器灰度

一个应用一般会部署多台机器，流量大的应用机器甚至50台以上，先上一台观察是否正常，验证通过后再上其他机器，这个过程叫机器灰度。这种灰度的优劣势如下：

优势	劣势
简单，几乎没有开发成本和学习成本。	①验证难度大，难以将验证过程的请求打到“新上线的机器”上； ②存在小规模故障风险，“新上线的机器”如果有bug，会小规模影响线上业务。

### •AB灰度

线上的代码成为A分支，待上线的代码成为B分支，通过请求中的某个参数（如用户ID等）与配置中心（如DUCC等）的配置进行比对，命中则走B分支，不命中则走A分支，这种灰度叫AB灰度。

伪代码如下：

 京东云开发者 

技术运营 @京东科技信息技术...

作者榜No.1

优秀作者

1.8k

文章

3.0m

阅读

19k

粉丝

关注

私信

目录

收起 ^

一、可灰度

二、可监控

三、可回滚

- 相关推荐
- ClickHouse在酷家乐日志监控系统中的...

3.4k阅读 · 42点赞
- 揭秘Java Agent技术：解锁Java工具开...

4.3k阅读 · 31点赞
- 全链路多重灰度发布

559阅读 · 12点赞
- 一文入门ETCD

21k阅读 · 80点赞
- 领域驱动设计（DDD）：软件设计的精粹

1.9k阅读 · 3点赞

- 精选内容
- 【设计模式】【行为型模式】命令模式（...

flzjkl · 33阅读 · 1点赞
- 用5分钟彻底掌握Python推导式（新手...

Asthenia0412 · 38阅读 · 0点赞
- 边缘计算框架HonoJs的权限控制

Y11\_推特同名 · 57阅读 · 0点赞
- Redis 持久化原理分析和使用建议

vivo互联网技术 · 82阅读 · 4点赞
- Java 类加载机制与堆区内存细分大揭秘

装睡鹿先生 · 60阅读 · 2点赞

找对属于你的技术圈子

回复「进群」加入官方微信群







▼csharp

代码解读

复制代码

```
1 .....
2 .....原代码.....
3 .....
4 void function(long userId){
5     /**
6      * 新需求，此处要求修改逻辑
7      */
8     call methodA();
9     do something;
10 }
11 void methodsA(){
12     do something
13 }
14 .....
15 .....新代码.....
16 .....
17 private List<Long> config = new ArrayList();
18 void function(long userId){
19     /**
20      * 灰度控制，仅指定的用户进入新逻辑
21      */
22     if(config.contains(userId)){
23         call methodA_new();
24     }else{
25         call methodA();
26     }
27     do something;
28 }
29 void methodsA(){
30     do something;
31 }
32 void methodsA_new(){
33     do something;
34 }
```

优劣势分析：

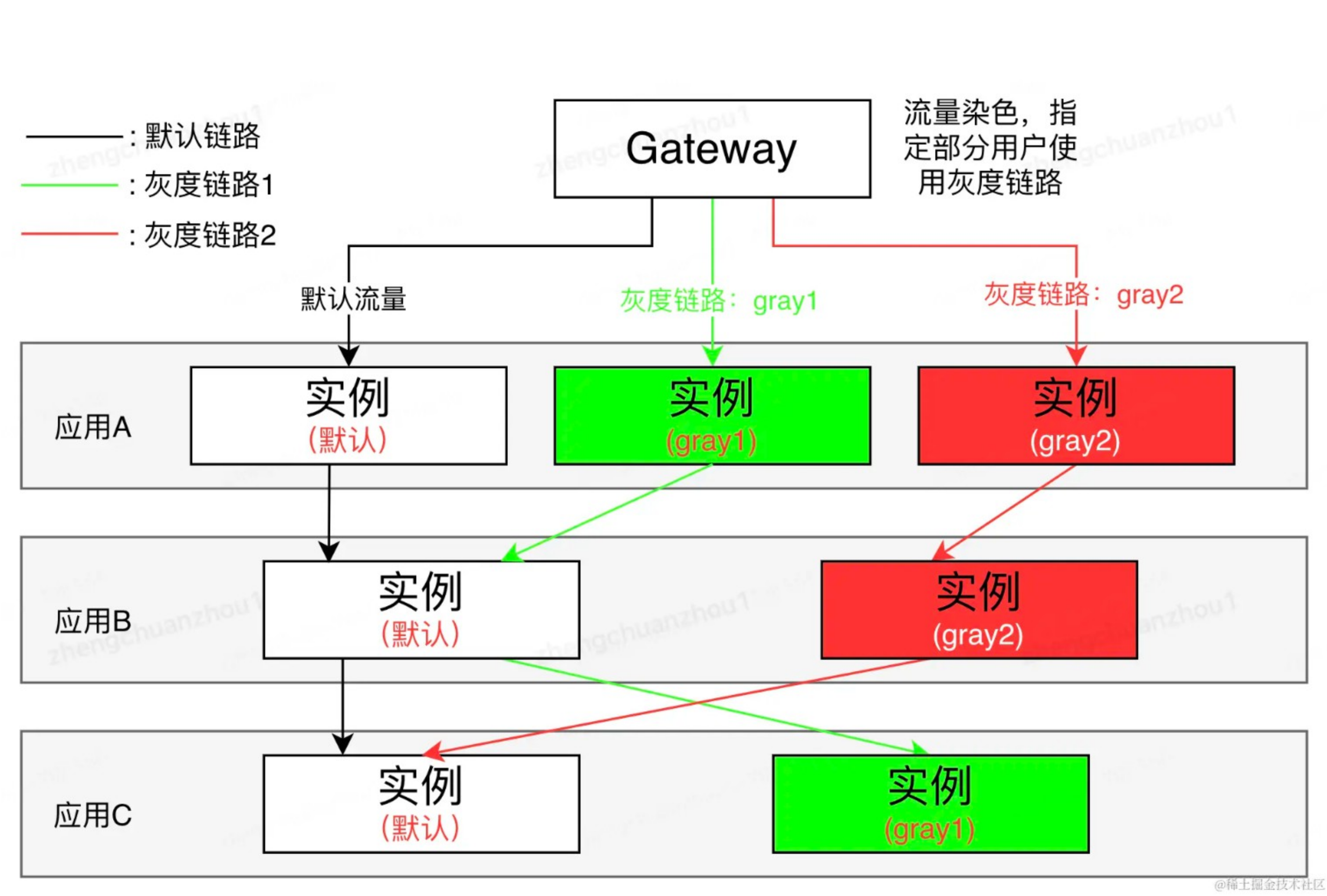
优势	劣势
⊙比较简单，5分钟就可以学会。⊙可以指定用户验证，先验证测试账户，再验证小流量商户，再逐渐放量，可避免线上故障。	⊙难以应对复杂需求，假设一个需求改动100个方法，通过AB灰度控制就显得很乱了；⊙需要上2次线，第一个加灰度控制，第二次去灰度控制。

其他重要说明：

AB灰度一定要控制好放量的节奏，而且每次放量都需要经历“自测”和“线上高峰期”验证，放量节奏的最佳实践是：测试商家 -> 1个商家 -> n个商家 -> 1% -> 5% -> 10% -> 30% -> 100%。

•全链路灰度

线上的代码是默认链路，新代码是灰度链路，给小部分流量染色，让染色流量走到灰度链路，这个过程叫全链路灰度。全链路灰度的示意图如下：



优劣势分析：

优势	劣势
⊙容易验证，by功能的灰度验证过程，研发能够很方便去验证是否符合预期；⊙代码侵入少，相对AB放量代码要简洁很多，也无需二次上线。	⊙需要依赖公司的基建能力，小团队难以搞定，比如对于京东，需要JSF平台、JMQ平台分别支持流量染色能力，以支持同步和异步调用。

其他重要说明：

在研发、测试过程中，经常发现环境不够用的情况，链路灰度的愿意应用到测试环境上，叫做“泳道环境”。举个例子：一个系统有100个应用，如果要部署5套环境，至少需要1005=500台机器，利用泳道环境，仅需要部署一套主干环境，用到100台机器，假设每个新环境仅改动到2个应用，那总机器数是100+24=108，大



大降低机器成本和时间成本。

•沙箱灰度

沙箱的概念在杀毒软件上用的比较多，杀毒软件会搞一个虚拟环境（类似虚拟机），然后将可疑文件放到虚拟环境中运行，然后分析是否有不安全的特征，协助判断是否是病毒、木马。

在互联网的应用上，变更是导致线上故障的元凶，因此，希望有个类似沙箱的环境，将线上流量引入到该环境进行验证（又不对线上造成影响），验证没问题后，再执行上线。

在百川系统上，大家做的AB对比，实际上就是沙箱环境，搭建了一个沙箱环境成为A环境，线上环境成为B环境，流量A环境、B环境运行的结果做对比，对比结果准确，才执行上线。

## 二、可监控

监控很重要，它是研发了解线上应用的窗口，其重要性可比眼睛对人的重要性。当系统监控不完善，研发就不能先于客户发现问题，研发的专业度就体现不出来，其实是一种耻辱。

对互联网公司来讲，监控不仅仅是机器监控，而是包括机器监控、链路监控、网络监控、业务监控等全方位监控，我们可根据应用的重要性配备上不同的监控系统，以便及时发现和处理线上故障。下表总结了京东在每个方向上存在的监控系统：

分类	说明	监控平台&官网	推荐
主机监控	针对机器的监控，包括cpu、内存、网络等	mdc、统一管控平台、brolly控制台、火眼	mdc
进程监控	监控jvm进行，比如CPU、FullGC、yongGC等	ump、pfinder、sgm	ump
性能监控	监控rpc性能，比如tp99、sla、调用量等	ump、pfinder、sgm、thor	ump
链路跟踪	根据tranceld查询到一个请求链路的所有节点，包括rpc调用，db等中间件调用等。	pfinder、sgm、cat	pfinder
日志监控	监控应用运行过程的日志	logbook、digger	logbook
网络监控	针对网络的监控	NP、DNP网络监控	NP
数据库监控	针对数据库的监控，包括cpu、load、磁盘利用率等指标	易维、CleverDB、noah平台、myops、DBS、数据库智能诊断系统、火眼	易维
中间件	针对redis、mq等中间件平台的监控，默认都需要开启并使用。	JSF、JMQ2、JMQ4、蜂巢、dap、clove、物流网关	每个中间件平台的监控，都需要使用
业务监控	针对业务的监控，比如可以监控订单的同比、订单未闭环、对账场景等。	前哨	前哨
安全监控	针对安全的监控	神盾	神盾
web端监控	针对web端的监控，比如监控方法的端到端性能、比如埋点等	DEEPLOG、子午线、烛龙、Watchtower、sgm-console-web、奇点	deeplog、子午线
app端监控	针对app端的监控，包括性能监控、crash监控等	SGM（移动端）、子午线、shooter、鹰眼 crash、奇点、烛龙	SGM（移动端）、子午线
大数据	针对大数据的使用和监控平台	dp、bdp、烽火台、imonet、京东动力	dp

对于监控来讲，核心指标是“漏报率”和“误报率”，而这2个指标是相互制约的，当把“漏报率”搞得极致则“误报率”会高，引发“狼来了”的现象，当把“误报率”搞到极致则“漏报率”会高，因此，需要平衡好这两个指标，可以这么定目标：误报率小于20%，漏报率小于20%。

## 三、可回滚

可回滚是兜底，谁也不能保障代码100%无bug，当线上出现问题时，“先止损，后查问题”是最佳实践，最快的止损方法是：灰度放量回滚，能够做到秒级恢复。但是，如果灰度放量回滚无法解决，那就需要触发兜底动作：回滚。

从个人情感角度出发，研发同学是不希望回滚的，但从客户角度出发，是不能接受线上长时间不可用，因此，可回滚是必须的，对线上的每一次变更，如果不能做到“可回滚”，是不可以上线的。

可回滚，绝对不是简简单单的操作应用回滚，而是要从多个角度评估是否可回滚、回滚后需要做什么，包括：

•应用回滚

应用是否可以操作回滚，如果涉及多个应用，回滚的顺序是什么等。

•数据库DDL回滚

数据库表的DDL是否需要回滚，如果需要，回滚时长多久、回滚期间会不会造成更大影响等。



•数据回滚

新逻辑产生的数据，回滚后是否兼容，是否需要修复数据，修数据的方案是什么等。


•其他

回滚还需要考虑哪些个性化的影响。

标签：

后端


评论 1



登录 / 注册

即可发布评论!

最热 | 最新

古曲 PHP工程师 @无

验证难度大，难以将验证过程的请求打到“新上线的机器”上；

有错别字

5月前 

👍 点赞

💬 评论

...

为你推荐

- 稳定性方法论：可灰度 & 可监控 & 可回滚

京东云开发者 | 1月前 | 👁 215 

👍 1

💬 1

Java
- 基于系统稳定性建设，你做了哪些事情？（下）

毕生狼 | 3年前 | 👁 3.2k 

👍 8

💬 评论

架构
- 稳定性治理思路

字节架构前端 | 6月前 | 👁 565 

👍 15

💬 2

性能优化
- 【从入门到了解】Android稳定性优化深入解析

程序员江同学 | 2年前 | 👁 5.0k 

👍 31

💬 3

Android
- 如何设计可靠的灰度方案

阿里云云栖号 | 3年前 | 👁 3.2k 

👍 19

💬 评论

数据结构
- 【稳定性】上线三板斧（可灰度、可验证、可回滚）

京东云开发者 | 2月前 | 👁 400 

👍 7

💬 1

架构
- 【稳定性】上线三板斧（可灰度、可验证、可回滚）

京东云开发者 | 2月前 | 👁 247 

👍 3

💬 评论

设计
- 我是如何保障亿级用户系统五年0故障

ali老蒋 | 1年前 | 👁 1.4k 

👍 18

💬 1

程序员
- 稳定性保障8个锦囊，建议收藏！

Vicla | 1年前 | 👁 663 

👍 9

💬 评论

运维
- 稳定性建设 -高可用系统建设的必备知识-下

梦尘啊 | 10月前 | 👁 527 

👍 3

💬 评论

后端

架构
- 业务团队如何在日常工作中做稳定性？涵盖事前、事中、事后的方方面面

Bella的技术轮子 | 3年前 | 👁 2.2k 

👍 12

💬 评论

架构

后端
- Kubernetes 稳定性保障手册：洞察+预案

阿里云云原生 | 3年前 | 👁 1.9k 

👍 4

💬 评论

Kubern...

后端
- 稳定性建设实践

木小丰 | 1年前 | 👁 794 

👍 2

💬 评论

后端

Java

代码规范
- 一种可灰度的接口迁移方案

阿里巴巴大淘宝技术 | 2年前 | 👁 2.2k 

👍 19

💬 评论

后端
- 稳定依赖原则（Stable Dependencies Principle）：构建健壮系统的黄金法则

肖哥弹架构 | 7月前 | 👁 69 

👍 3

💬 评论

后端

架构

Java