CODE A BETTER LIFE
一 行 代 码　　亿 万 生 活

美团扫码付前端可用性保障实践

分享人 田泱

美团

有田十三。

Chaoyang, Beijing

Scan the QR code to add me on WeChat

田泱

智能支付 用户前端组 看护工程师

2017年1月，开始负责扫码付前端

2016年5月，加入金融服务平台

2015年7月，加入美团

抛砖引玉的话

2018年上半年 前端回顾：

1. PWA
2. 小程序
3. Flutter
4. ？？？

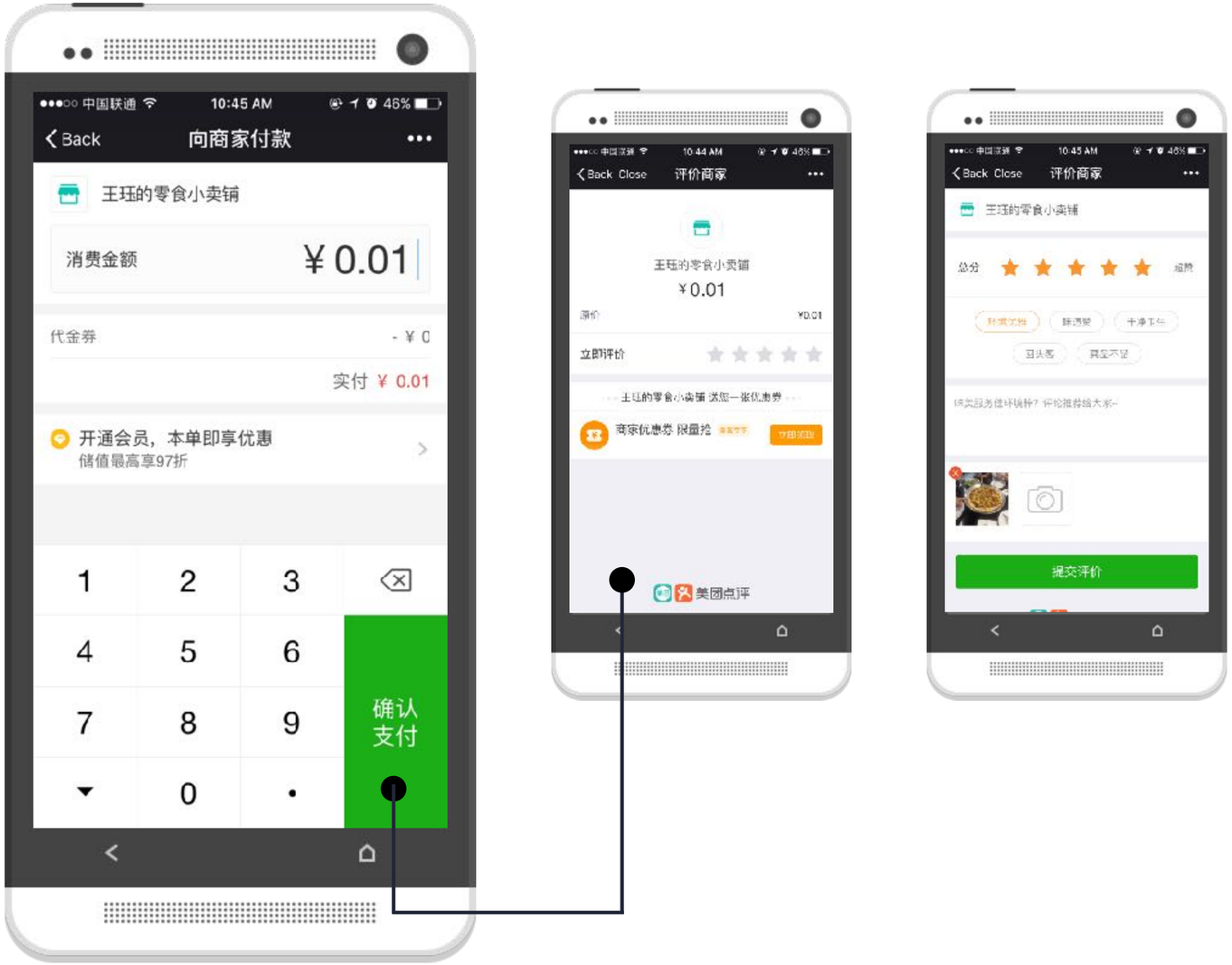
目录

- 扫码付前端可用性定义
- 影响可用性的关键因素
- 端到端监控与降级
- 保障动作标准流程

业务介绍

二维码聚合支付

二维码支付是一种聚合支付产品，美团金融线下收单产品之一，为商户提供聚合支付的收银解决方案。



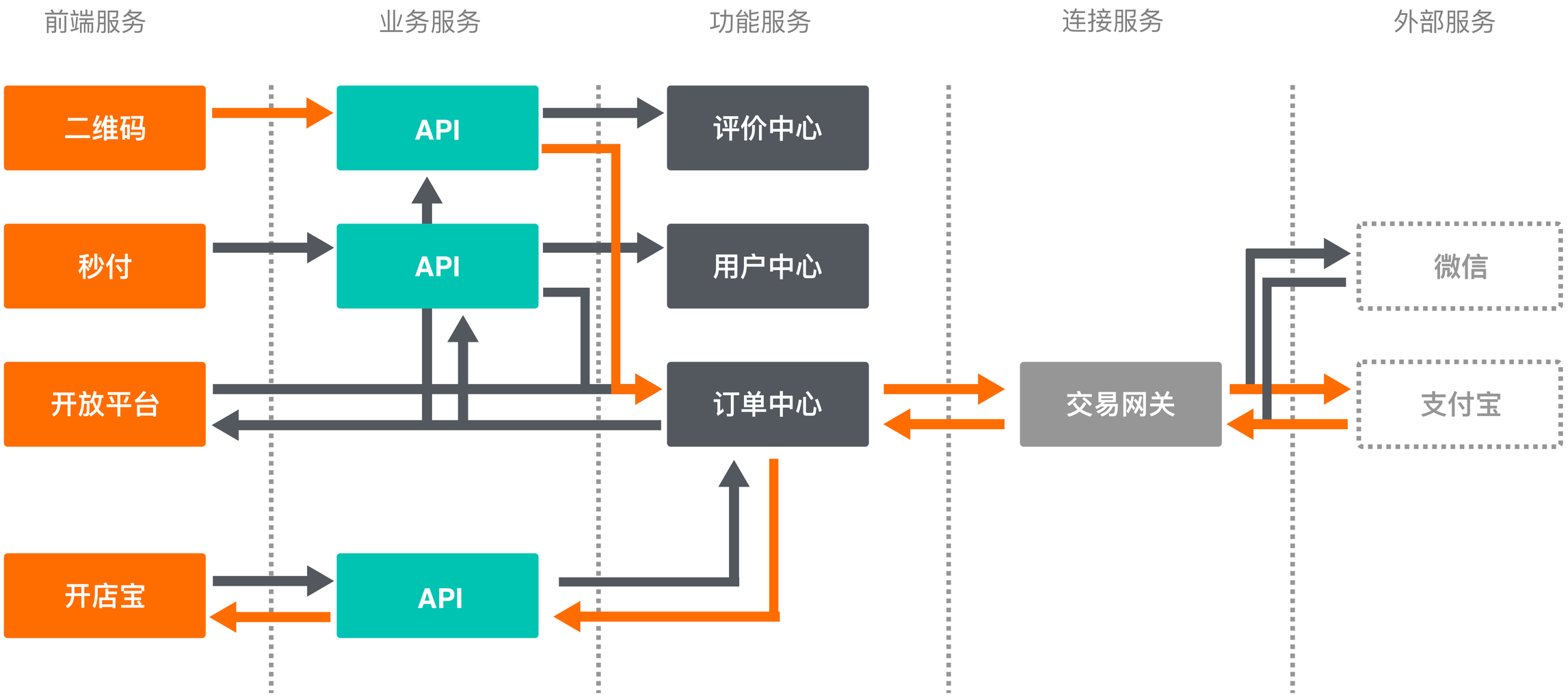
搭载火箭的冰山式服务。

业务介绍

理想情况

现实情况　随着业务发展，核心链路上功能节点越来越多，功能点越多，故障点也越多

业务介绍


前端服务
业务服务
功能服务
连接服务
外部服务
二维码
秒付
开放平台
开店宝
API
API
API
评价中心
用户中心
订单中心
交易网关
微信
支付宝

不得不承认的事实

“ mmp，为什么要做前端？ ”

你认为你的前端服务

实际你的前端服务

“ 我们要提升对服务的信心，才能心安理得地面对你想要的诗和远方！ ”

~~“ mmp，为什么要做前端？ ”~~

扫码付前端服务可用性保障实践

扫码付前端可用性定义

系统可用性 = 无故障时间 / (无故障时间 + 故障时间) * 100%

- **要么生，要么死 的后端可用性：0 || 99.99+**
- **生死不如 的前端可用性：[0 , 99.99 ）**

思考问题

· **提升万分之一可用性的价值？**

扫码付前端服务可用性保障实践

影响可用性的关键因素

历史故障回顾：

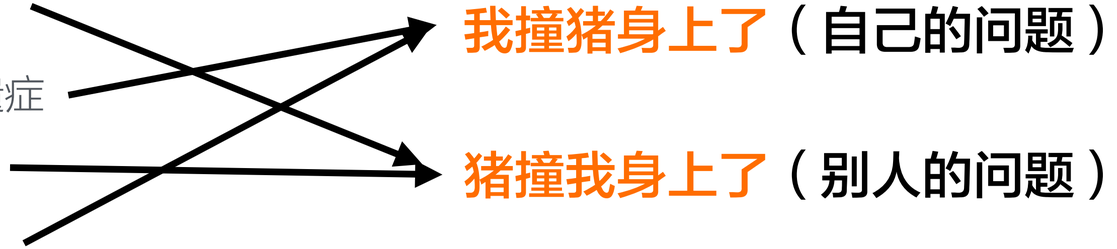
1. 客户端升级兼容性问题
2. 代码优化、服务迁移后遗症
3. 外部依赖服务故障
4. 研发流程执行不彻底

扫码付前端服务可用性保障实践

影响可用性的关键因素

历史故障回顾：

1. 客户端升级兼容性问题
2. 代码优化、服务迁移后遗症
3. 外部依赖服务故障
4. 研发流程执行不彻底



我撞猪身上了（自己的问题）

猪撞我身上了（别人的问题）

影响可用性的关键因素

内部节点可用性

“ 我撞猪身上了。 ”

保障服务可用的基本原则ABC

A 流程规范

- 研发手册
- 安全检查
- 上线准入标准
- 测试流程

B 方案合理

- 设计从简
- 避免黑盒

C 代码规范

- 更严格的语法ts
- 强制的eslint
- 严格的code review

影响可用性的关键因素

内部节点可用性

保障服务可用的基本原则ABC

B

方案合理

- 设计从简
- 避免黑盒

选型	优势	不足	适合
后端为主MVC	搭建简单、快速	前后端没分离、需要整套搭建环境、本地开发不易	缺乏专业前端、前端变化不大项目
基于NodeJS的前后端分离	前后端分离、前端范围扩大、控制力强	需要额外维护Node层、技术栈要求较高	缺乏服务端渲染、性能有所影响
纯静态化开发模式	最小代价的前后端分离	缺乏服务端渲染、性能有所影响	小型轻量项目、中后台系统

影响可用性的关键因素

内部节点可用性

保障服务可用的基本原则ABC

C

代码规范

- 更严格的语法ts
- 强制的eslint
- 严格的code review

```
// 函数类型被覆盖
let a = 1,
    a = 'abc';

// 对象成员属性不明确
let obj = {
    a: 123,
    b: 123
};
console.log(obj.c);

// 函数返回的类型被隐式转换
(x => x + 5)('1')
```

影响可用性的关键因素

内部节点可用性

更多其他保证手段

D

测试效率

- 单元测试覆盖率
- 自动化UI测试

01 自动化测试的意义

- 降低测试成本，提高迭代效率
- 定期容器版本兼容测试，降低容器升级风险

02 自动化测试的难点

无法模拟微信容器，无法模拟真实支付场景

03 现在的解决方案

- 通过**Appium**可实现安卓端微信下h5页面自动化测试
- Appium控制设备启动微信，按测试用例模拟点击
- 对比预设结果与实际结果，输出测试报告

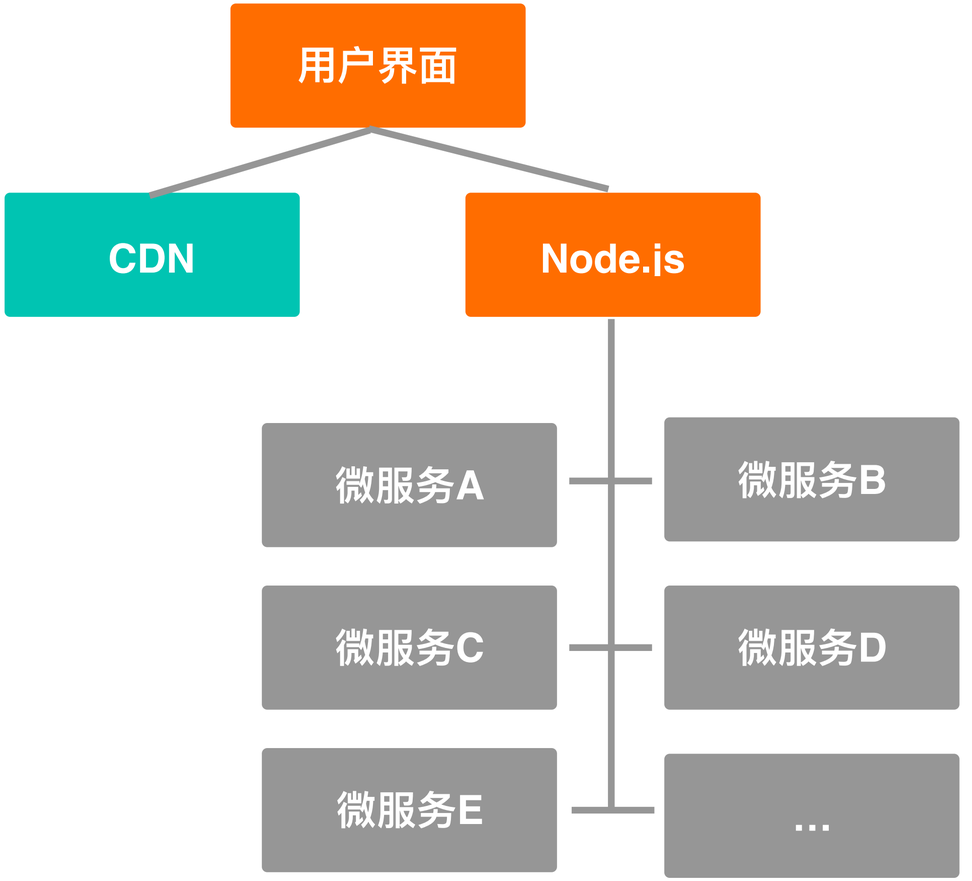
影响可用性的关键因素

外部链路可用性

“ 猪撞我身上了。 ”

简单可依赖，不存在的

系统可用性 = 资源可用性 * 接口可用性



影响可用性的关键因素

外部链路可用性

> “ 猪撞我身上了。 ”

资源不可用分析

01 网络抖动/异常

因为网络环境不稳定造成的网络不可用，表现为文件加载失败

02 网络劫持/代码篡改

运营商劫持，表现为篡改代码，注入广告

03 代码执行错误

代码执行失败，多为执行环境代码不兼容，或者CDN节点异常

影响可用性的关键因素

网络抖动/异常 解决方案

CSS域名重试

- 失败会切换CDN域名进行重试
- 每天约2000次重试，70%重试命中

CDN降级回源

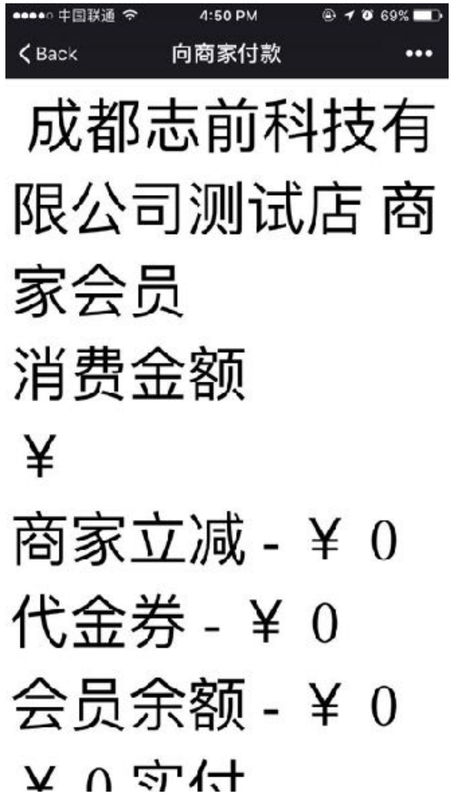
- 核心文件加载失败，回源降级成mvp
- 非核心文件读取ls缓存保证正常加载
- 每天约1000次降级，补充订单100+

思考问题

- **服务是否可以承受大面积回源?**

正常情况，会员门店

未降级前，大字报



降级后，普通门店

影响可用性的关键因素

网络劫持/代码篡改 解决方案

劫持原因

- 目的是**节省跨运营商及跨省节算的带宽**
- 运营商**独立运作**，无法通过集团进行整体要求
- **劫持缓存**，移动并不会主动解除

解决方案

· HTTPS全覆盖
· 120/Doctor模式
· 省/市级区域 资源性能监控

思考问题

· **CDN回源请求是否使用HTTPS？**

影响可用性的关键因素

外部链路可用性

“ 皮之不存，毛将焉附 ”

接口不可用分析：

01 网络失败

因为网络环境不稳定造成的网络不可用，或者接口响应超时

02 业务异常

参数错误，业务逻辑处理错误，返回数据格式错误等下游服务不可用

03 机器故障

CPU负载过高、内存溢出、磁盘打盘

解决方案： 监控，快速反馈

端到端监控与降级

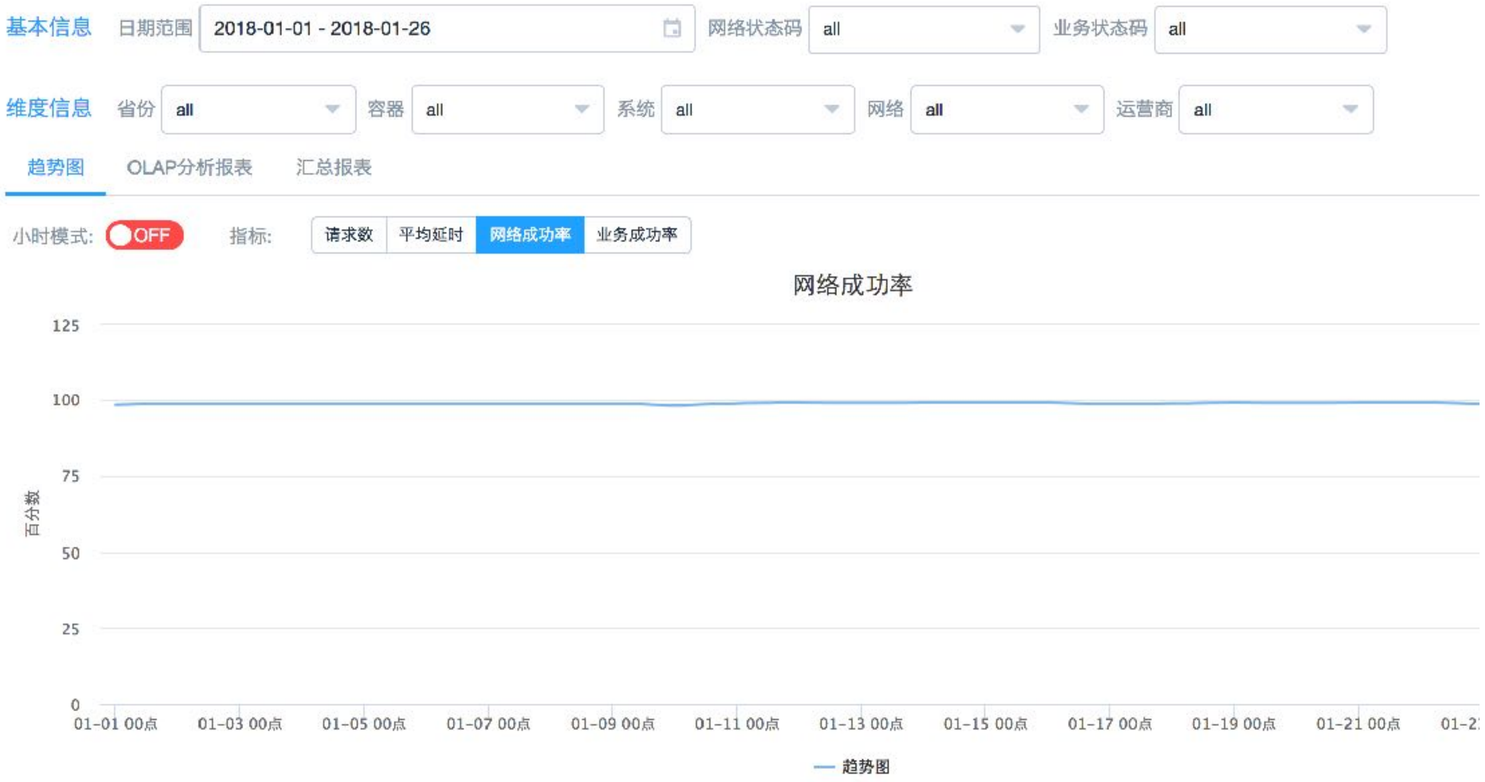
监控：正面的预防措施，快速发现问题进行报警；

降级：反向的止损措施，遇到故障自动处理，降低损失

监控项	预防	解决方案	工具	注意事项
机器	内存、QPS、磁盘使用率	双机器备份、双机房容灾 自动摘除节点	OCTO、Falcon	
服务	服务异常、崩溃	进程守护	PM2	
接口	网络异常、业务错误 下游服务不可用	异常文案引导	logcenter、大象	
资源	代码篡改 代码执行错误	错误信息堆栈回收	cat	省/市维度
流量	流量激增/骤减 DDOS、DNS劫持	安全伞、HTTPS	cat	省/市维度
性能	DNS劫持	HTTPS	cat	省/市维度

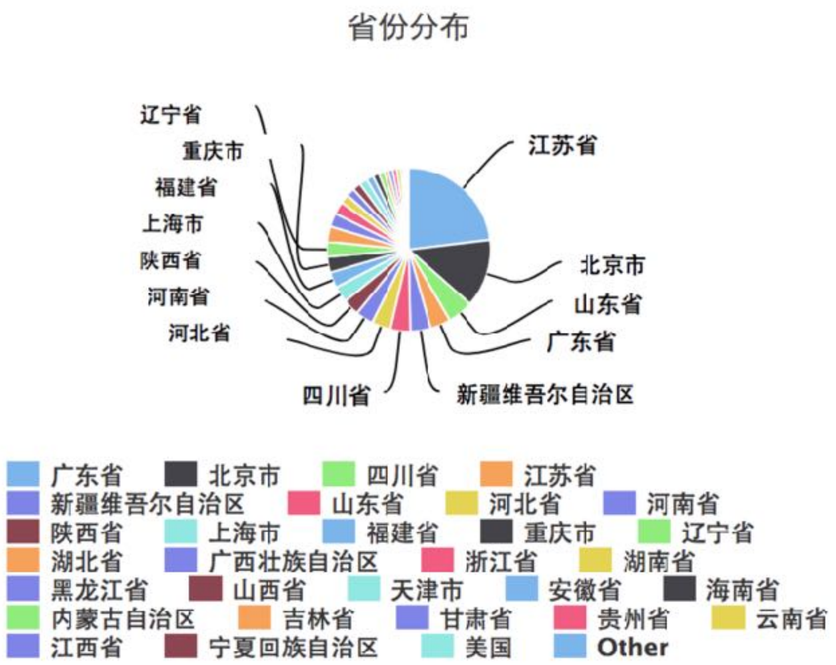
端到端监控与降级

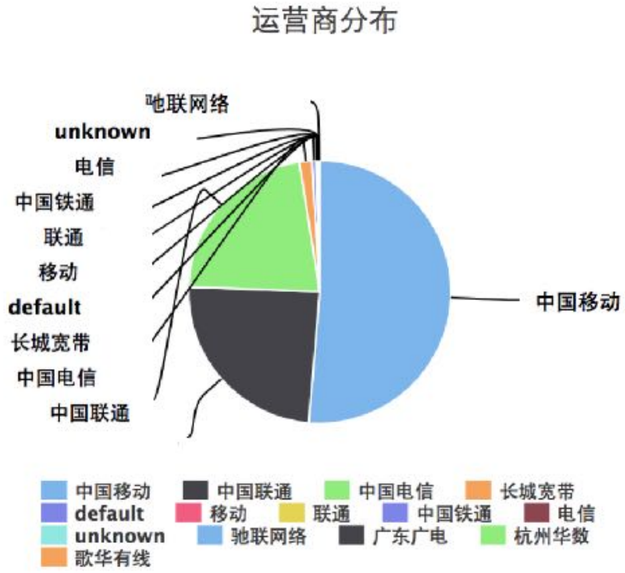
Cat：实时业务监控平台



端到端监控与降级

CAT(Central Application Tracking)：分布式实时监控平台





端到端的监控与降级

故障演练的必要性

“ 实践出真知 ”

如何快速提高对系统的信心？

- 检查有效性，降低方案影响范围
- 提高熟练度，降低故障影响时间

扫码付前端服务可用性保障实践

保障动作标准流程

01 事前，依赖流程与规范

方案设计原则：合适从简的基础上，尽可能避免核心链路的黑盒

线上准入SOP：代码检查、CodeReview、测试标准、应急预案

02 事中，依赖监控与降级

接口、资源、流量、性能、服务、机器等监控与报警，尽量做到可以全部自动降级；

无法降级的问题要有应急预案，通过演练与培训提高熟练度；

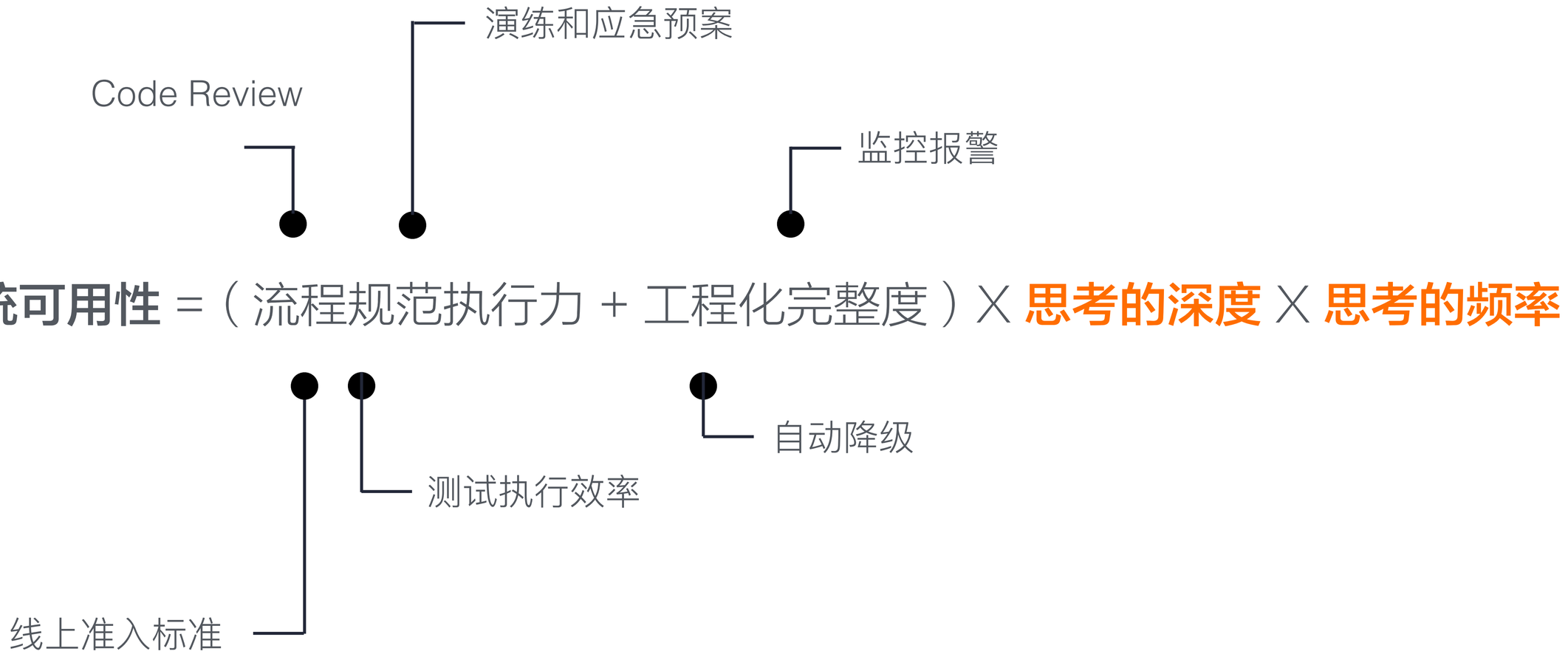
03 事后，依赖执行力

Case Study

扫码付前端服务可用性保障实践

总结

“ 让开发变得更简单。 ”



扫码付前端服务可用性保障实践

总结

“ 让开发变得更简单。 ”

线下一站式的开发环境

自动化构建部署

运行时托管

解放精力

Erajs + Ones = Techless Web Apps

自助运维监控

全生命周期的
自定义插件生态

企业级服务集成

专注业务

Q&A

扫码关注美团点评技术团队公众号
获取最 IN 的技术资讯

招聘：高级前端工程师 / 前端技术专家
邮箱：tianyang02@meituan.com

美团点评 | 技术团队