WOT
World Of Tech

2016年8月26日-27日　北京珠三角JW万豪酒店

World Of Tech 2016
移动互联网技术峰会
THE BEST MOBILE TECH IN HERE

主办：51CTO

故障与高可用

张明强

对于维护者来说

99%的故障诱因

都是**极其简单**的

故障与高可用

- 故障诱因
  - 问题形式
  - 解决方法
- 解决方法
  - 核心思路
  - 易踩的坑

买饭记

外出买饭，下雨被淋

回家换衣

网上订餐

支付前发现没有饮料

重新点餐，支付

快递员送餐上门

三大问题

挂

1. 彻底消失 – 最理想的状态
   - 断电 – 最常见
   - 光纤被挖断
   - kill -9

2. 假死
   - GC过长
   - 主流程死锁
   - 硬盘故障IO卡死

3. 闪断 – 最麻烦
   - 网卡切换
   - Redis异常bgsave

机器宕机
交换机上联模块故障
exception crash
内存爆掉
服务器断网
淘宝故障
Java OOM
内存错误-宕机
运营商屏蔽443端口
DNS劫持

挂 – 解决方法

1. 监控 – 外出买饭，**我发现**衣服被淋
   - 发现问题，才能解决问题。

2. 多实例 – **换一件**干净衣服
   - 一个人倒下去,千万人顶起来

3. 拆分 – 电脑没被淋湿（打开电脑订餐）
   - 业务拆分，防止一挂全挂

慢

1. 自身慢
   - 逻辑BUG
   - CPU Load升高

2. 上游慢 – 易被忽略
   - 网络异常
   - 上游发送请求数据过慢

3. 下游慢 – 易与自身慢混淆
   - 数据源
   - 网络异常

NodeJS只用一个CPU
Redis慢
QPS暴增
CDN异常回源
容量评估错误
切换机制过长用户退出
访问国外网络超时
无索引查询
同步依赖监控系统
消费端不够队列积压
CPU Load
流量堵塞网络丢包

慢 – 解决方法

1. 多实例（并发）
   - 对于既定的体力劳动，人多力量大

2. 缓存 – **在家**网上点餐
   - 近水楼台

3. 拆分（异步） - **快递员送上门**，我在家等着
   - 流水线工作

错

1.错 !=

- 不一致
- 改错

2.空 NIL

- 误删

代码BUG删错数据
广告逻辑BUG
锁使用错误数据不一致
切换后故障机仍提供服务
代码重构未改全
配置文件写错
使用bitmap出错
MHA主从切换错误
代码写错语音上传失败
搞错主从进行操作
Jackson升级后解析错误
页面Js没有写完整
修改字段未同步给客户端
忘了给token续期
+号解析错误
新增字段未设置default
新机器没有同步代码

错 – 解决方法

1. 流程规范 – 在支付前重新检查了一遍订单
   - 技术不是万能的

2. 拆分
   - 不同业务物理隔离，防止一错全错

3. 多实例
   - 错了后，有备份数据恢复

五种方法

监控

没有监控，就是瞎子。

形式

- 业务
- 平台/框架
- 硬件

易踩的坑

1. 监控 = 数据 + 报警。不要有数据没报警。
2. 报警是要看的。
3. 靠监控自动处理的设计，小心误报，要依赖多因素验证。
4. 监控拖垮服务。

监控 –监控什么

以用户注册为例

1. 目标是否达成

- 注册总数
- 注册失败
- 性能

2. 是否可以细分

- 手机注册
- 邮箱注册
- 微信注册
- 注册每一个步骤的量

3. 易出问题地方

- 短信/邮箱通道监控
- 微信接口监控

4. 业务外系统监控

- CPU
- 网络
- 数据源

多实例

有备无患

形式

- 冷备（被动切换）
- 热备（自动切换）- Keepalived
- 多活（自动切换）- 微服务

易踩的坑

1. 会不会同时挂掉？
   - 常见：在同一个机器、机柜、交换机下
2. 客户端会不会切换失败？
3. 备机容量够不够？

多实例

监控项	个数	总容量	单个健康状态	同步状态	部署状态
任务分发	人工触发	备机检测	Master调度	客户端调度	
表现形式	冷备	热备	多活		

拆分

没有拆分解决不了的高可用问题

形式

- 拆入口（主从、Upstream、域名）
- 拆阶段（异步、管道）

易踩的坑

1. 只从产品业务上拆，而没有考虑技术指标
    - 快的与慢的混在一起

拆分

协调方式

接口

共享数据源

公共代码库

表现形式

入口

阶段

缓存

近水楼台先得月

形式：

- 数据缓存（OPCode也是数据）

易踩的坑

1. 无命中率监控，极低而不自知。
2. 脏数据。
3. 系统对穿透率支持的太低。
4. 缓存太散，导致IO次数太多，耗时太长。

缓存

访问策略	单机	取余	一致性Hash	其他	
淘汰策略	不淘汰	LRU	LFU	FIFO	
回填策略	R/W Through	代理模式	定时刷新		
关键监控	Hits	Miss	命中率	容量	性能

流程规范

最后的屏障

形式：

- 强制规范
- 倡导类的（相当于没用）

易踩的坑

1. 一大堆规范。
2. 太复杂，难以执行。
3. 无人遵守，管理上不跟进。

流程规范

对于故障

上线，最重要的一个规范

灰度，最重要的一个环节

最后

春来：基于异常的架构设计

三大问题

- 挂
- 慢
- 错

通用解决方法

1. 监控
2. 多实例
3. 拆分
4. 缓存
5. 流程规范

Thank you

微博：@寸谋
邮箱：4416@cunmou.com