程序员修炼
——避免千里之堤溃于蚁穴

KECO袁文科

2017年05月

关于我

目录

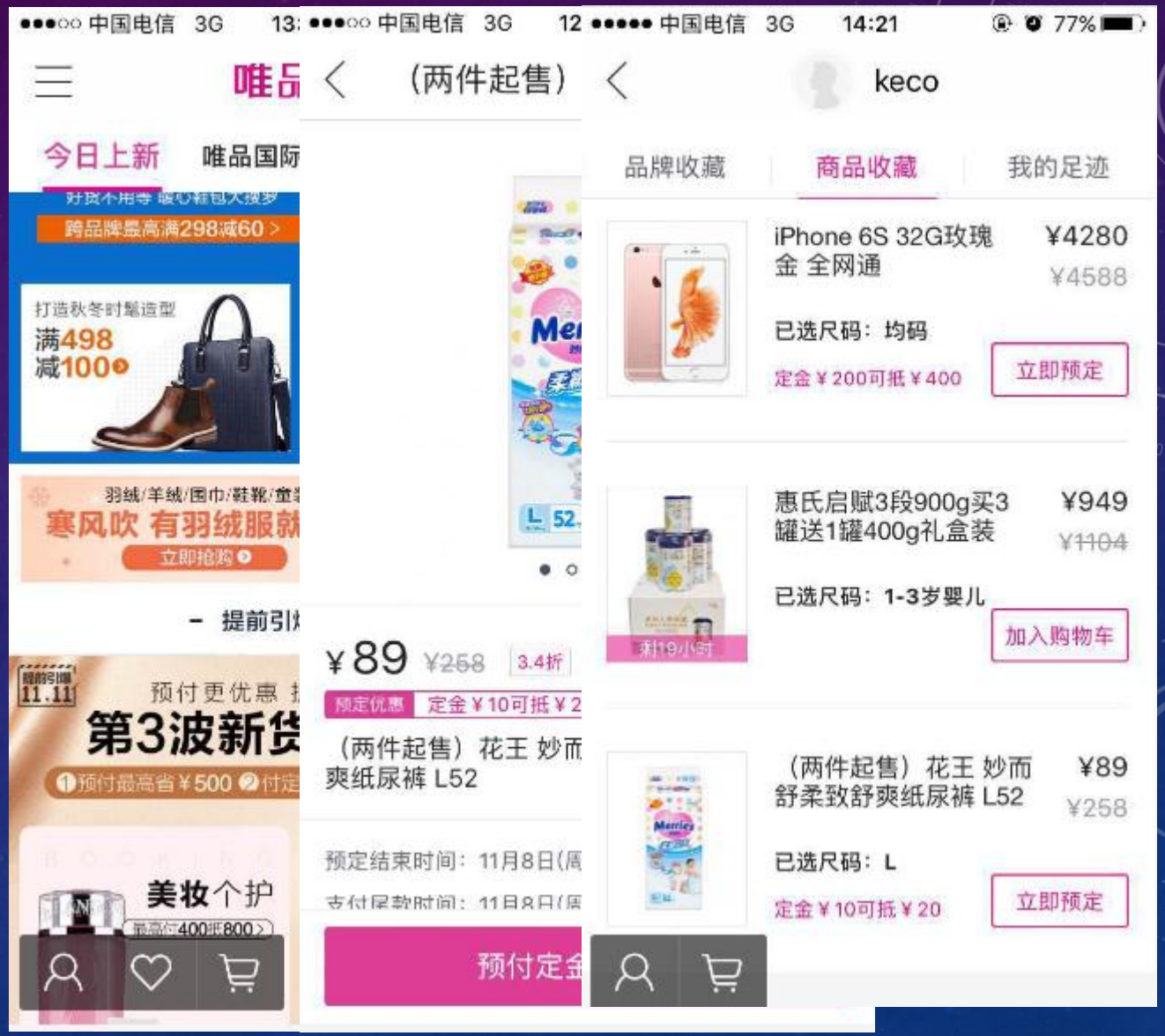
01 业务简介

特点、流程、核心举例

业务简介-特点

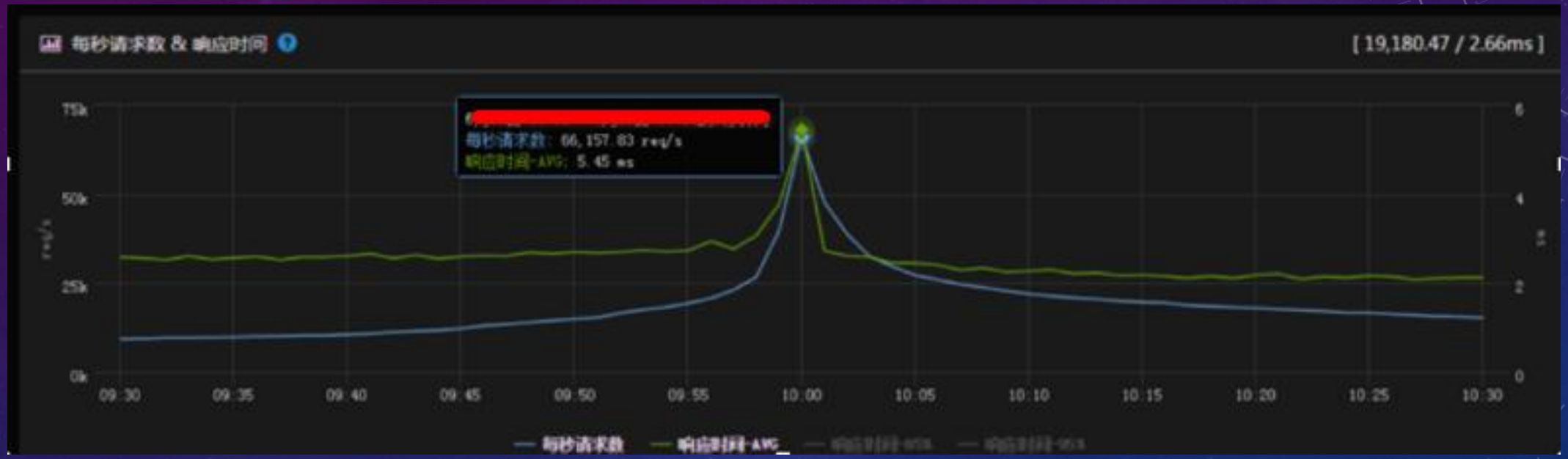
一家专门做特卖的网站

精选商品
深度折扣
限时特卖



业务简介-流程

业务简介-收藏



瞬间10倍流量

02 必经的历练

系统发展过程

历练-收藏一

- 易超时（4-5个域）
- 1：N转化请求致底层系统压力大

历练-收藏二

- 大促前全量缓存，冗余大（用户维度）
- 手工预热，不可控
- 全量缓存无法感知更新，数据变，报障来

03 迎接挑战

突破、设计、性能

挑战-突破点

综合问题

- 流量陡增
- 日常和大促方案各一，容量无预知
- 逢大促忧心忡忡、焦头烂额

挑战-设计

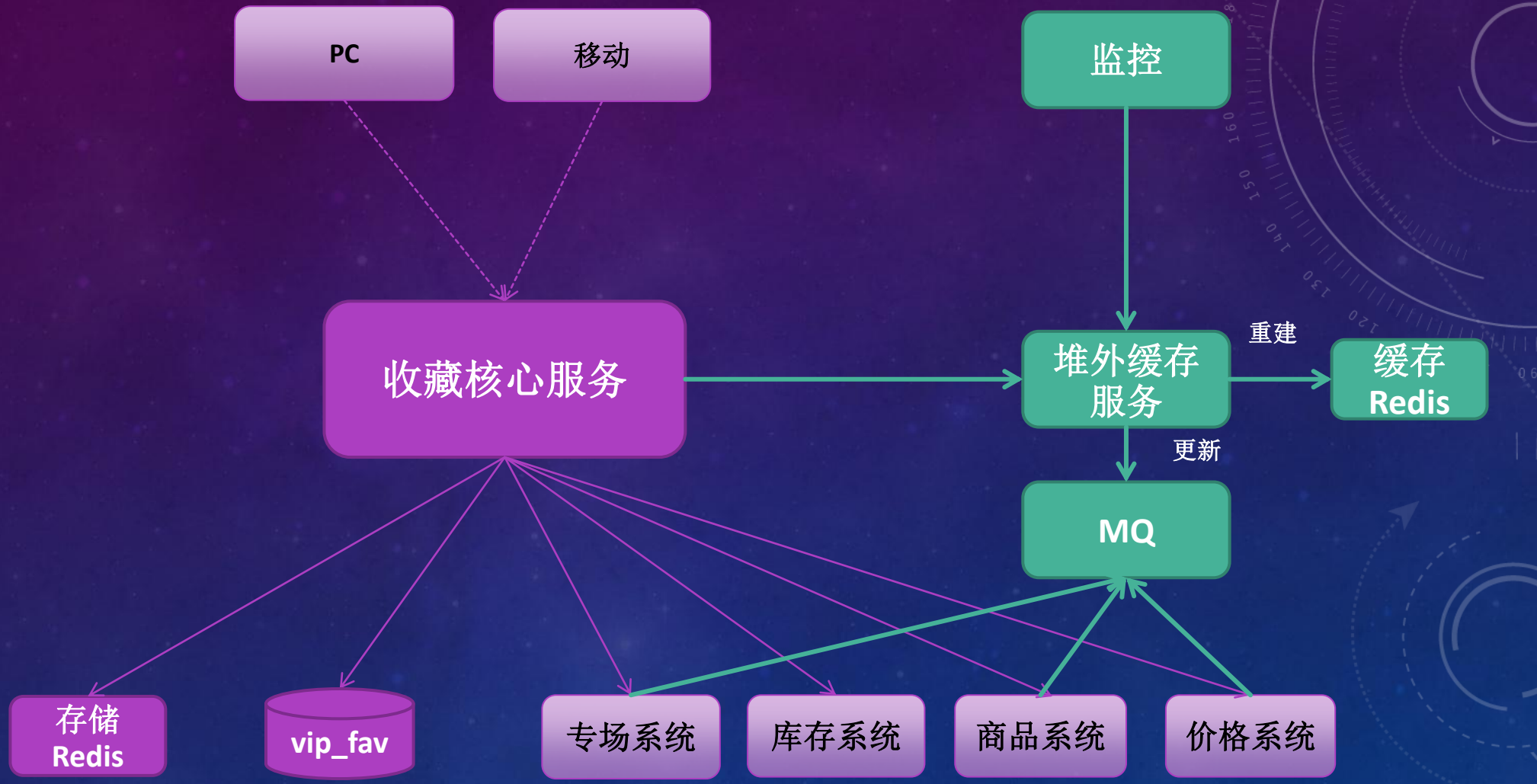
数据结构：用户Id维度——>专场、商品维度
存储空间：>6G
变化：频繁

分布式业务痛点

- 批量查询
- 热键
- 可靠性

类别	吞吐量	序列化	GC影响	适合场景
堆内	高	无	大	数据量小，更新不频繁
堆外	低	有	无	数据量大，更新频繁

挑战-设计


PC
移动
监控
收藏核心服务
堆外缓存
服务
重建
缓存
Redis
更新
MQ
存储
Redis
vip_fav
专场系统
库存系统
商品系统
价格系统

04 蚁穴之殇

雪崩、N倍、穿透、死锁

2011.7.23温州高铁事故

2015.5.28携程宕机事故

一段代码引发的血案……

蚁穴之殇-雪崩


PC
移动
WAP
https
F5
登录授权
LVS
支付短信
Token DB
Twemproxy oauth
Twemproxy wapsession
Twemproxy ……
Token Redis
缓存
缓存

蚁穴之殇-N倍放大

蚁穴之殇-N倍放大

```
JedisPool extends Pool<Jedis>
```

host:port

```
ShardedJedisPool extends Pool<ShardedJedis>
```

host1:port2
host2:port2
host3:port3
……

蚁穴之殇-N倍放大

JedisPool

```
<property name="maxIdle" value="
<property name="maxTotal" v
<property name="maxWaitMil
<property nam ="testOnBorr
<property name="jmxNamePre
```

```
@Override
public boolean validateObject(PooledObject<Jedis> pooledJe
final BinaryJedis jedis = pooledJedis.getObject();
try {
    return jedis.isConnected() && jedis.ping().equals("PON
} catch (final Exception e) {
    return false;
}
}
```

Jedis-2.4.2.jar

```
}
if (p != null && getTestOnBorrow()) {
    boolean validate = false;
    Throwable validationThrowable = null;
    try {
        validate = factory.validateObject(p);
```

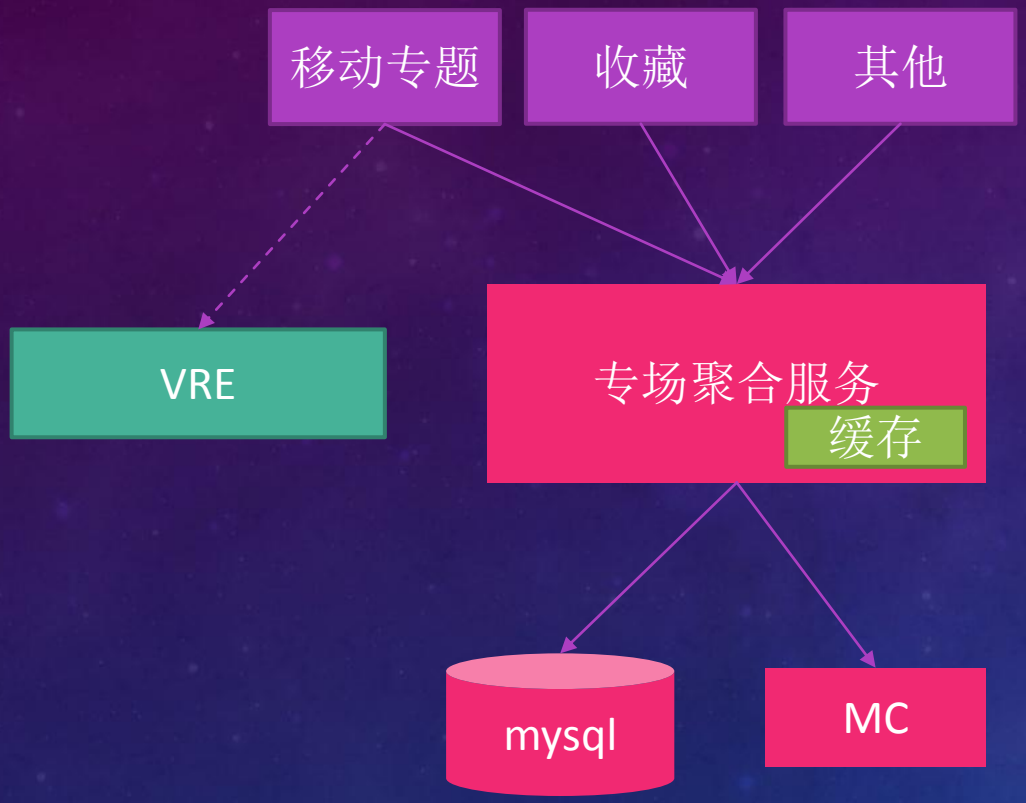
ShardedJedisPool

```
@Override
public boolean validateObject(
    PooledObject<ShardedJedis> pooledShardedJedis) {
    try {
    ShardedJedis jedis = pooledShardedJedis.getObject();
    for (Jedis shard : jedis.getAllShards()) {
        if (!shard.ping().equals("PONG")) {
            return false;
        }
    }
    return true;
    } catch (Exception ex) {
    return false;
    }
}
```

蚁穴之殇-穿透

	last	min	avg	max
[avg]	16.77 K	10.89 K	93.02 K	191.56 K
[avg]	0.1825	0.1582	0.1805	0.1925
[avg]	3.36	0.05	33.56	296.75
[avg]	8.53	0.2583	76.72	618.42

蚁穴之殇-穿透



原因

1）追溯主专场的数据缓存失效，大量回源

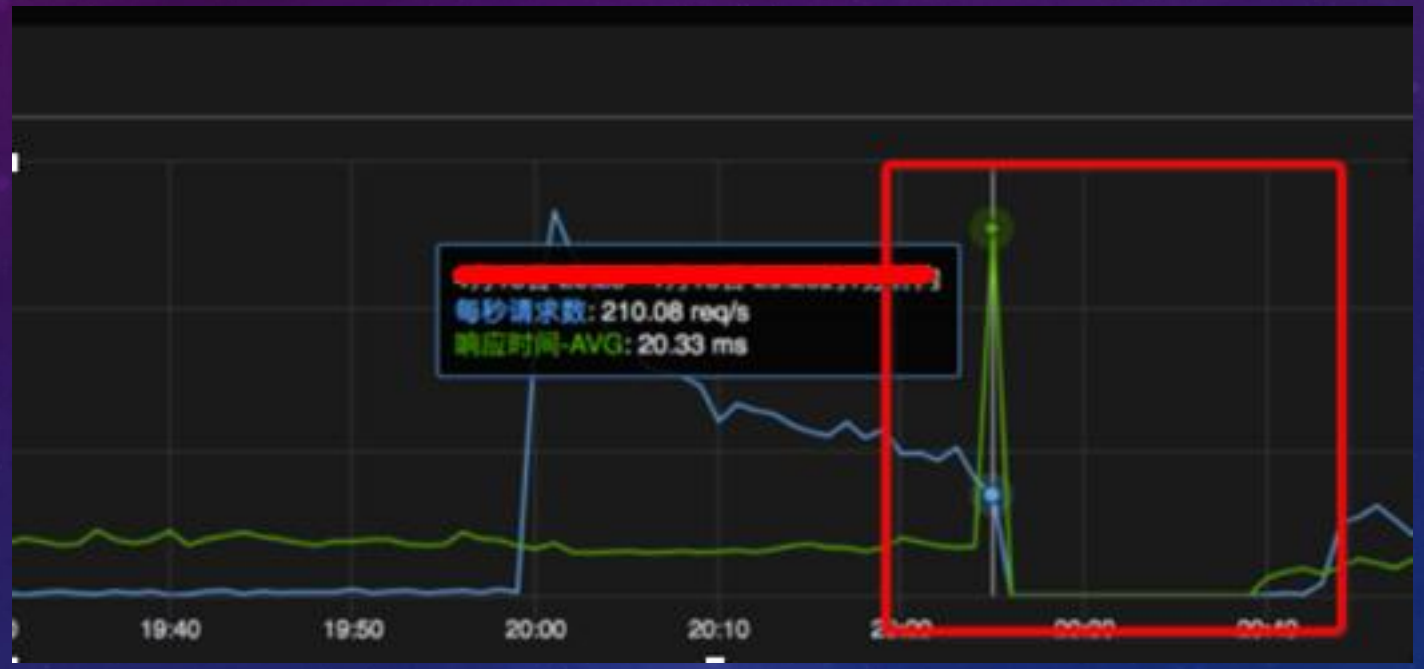
2）本地缓存调整半小时后发现哨兵机制刷缓存

存在漏洞，刷新缓存时没更新缓存时间导致不断

触发哨兵机制刷新

3）大量商品没有在预热范围内，导致db穿透

蚁穴之殇-事务死锁



蚁穴之殇-事务死锁

1、创建订单请求A系统

1.1 A系统请求B系统写入数据

1.2 A系统其他操作请求

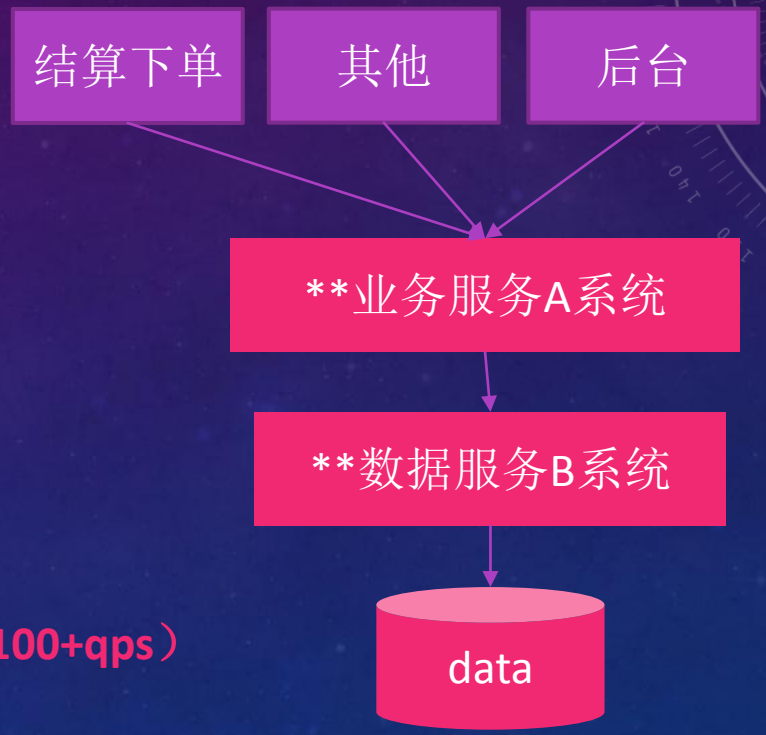
1.3 A操作派券

1.3.1 检查活动下线

1.3.2 写入异步队列

1.3.3 异步消费队列

1.3.3.1 请求B系统同步下线更新（8:20开启100+qps）



蚁穴之殇-事务死锁

B系统下线更新：

事务A：更新状态表

事务B：更新索引

事务嵌套导致连接占满

数据库连接等待死锁

外部请求占用容器连接等待

容器连接占满，服务挂死

```
@Override
@Transactional(propagation = Propagation.REQUIRES_NEW, rollbackFor = Throwable.class)
public boolean batchUpdateIndex(Byte type, List<String> actNoList) {
    if(!isOpen()){
        if(LOG.isInfoEnabled()){
            LOG.info("[batchUpdateIndex] active index has been shut down.");
        }
        return true;
    }

    Map<String, Set<String>> keysToUpdate = new HashMap<>();//需要更新的key
    List<String> keyToDelete = new ArrayList<>();//需要删除的key
```

05 修炼之道

设计、实现、检查

修炼之道-设计

- 合理业务逻辑拆分
- 合理系统分层设计
- 系统容灾机制、应急备案
- 简化设计，越简单越好

修炼之道-实现

- 切莫急于coding，多做思考和设计
- 深入了解配置意义和合理优化设置值
- 依赖服务故障情况下的处理和验证
- 极端情况下的补救机制（是否有办法补救）

修炼之道-检查

- 定期检查系统和数据库
- 及时review代码和变更

制定《清单》和执行实践

执行流程：开发填单[redacted]IRE TEAM共同约定执行
给开发者—>开发完成单提merge请求—>master合并更新单给测
测试验收上线—>返回给master

分类	检查项	对接人	确认结果
提测前	是否打印如下表格？	开发	
	单元测试是否完成通过	开发	
	A、B角是否达成一致？(设计，规范等)	A、B角色	
测试前	是否案例评审review通过	测试	
回归前	是否回归历史版本环境正常？	测试	
测试	是否回归hosts配置正确？	测试	
	是否回归环境变量配置正确？	测试	
	前端测试不清缓存案例执行是否ok？	测试	
	是否日志输出正常？	测试	
开发	是否有属性文件变化？	开发	
	是否有url请求参数增加	开发	
	是否有配置文件变化？	开发	
	是否有ddl或者dml脚本？	开发	
	是否有error基本信息？	开发	
	是否上传release客户端？	开发	
	是否提交sql的ccs申请？	开发	
	是否提交配置的ccs申请？	开发	
	是否静态文件（js）变更？	开发	
	是否B角色沟通检查？	开发	
	是否安全评审完成？	开发，测试	
	是否旧接口属性变更或者新增回归范围？	开发，测试	
	是否需要确认上线步骤？	开发，测试	
	是否影响客户端？	开发	
	是否有回滚方案？	开发	
	是否有修正release版本号？	master	版本号：
	是否代码review？	master	
	是否与上面的问题回答一致？	master	

修炼之道-莫存侥幸

“墨菲定律”：
如果坏事有可能发生，不管这种可能性有多小，它总会发生，并且可能引起更大的损失。

定律启示:
做任何事都要考虑到最坏情况，不要抱有侥幸心理。

THANK YOU

下期预告：《堆外缓存探讨与应用》