极客时间 训练营

极客时间 Java 进阶训练营
第 28 课
分布式系统架构--如何设计一个秒杀系统

KimmKing

Apache Dubbo/ShardingSphere PMC

个人介绍

Apache Dubbo/ShardingSphere PMC

前某集团高级技术总监/阿里架构师/某银行北京研发中心负责人

阿里云 MVP、腾讯 TVP、TGO 会员

10 多年研发管理和架构经验

熟悉海量并发低延迟交易系统的设计实现

极客时间训练营

目录

1. 从架构师视角出发

像架构师一样思考问题

为什么要做秒杀？

为什么秒杀难做？

第一性原理：秒杀的本质到底是什么？

秒杀的本质

业务上：一场营销促销活动，具有明确的活动业务特点

技术上：一种主动 Ddos 攻击，具备技术的不确定性和复杂度

早期的秒杀怎么做

早期的单体系统，不具备很好的扩展性，一有突发流量就挂。

那么如何实现抗住较大的突发压力呢？

最开始大家是真不知道，摸石头过河。

大促宕机时常态。

技术上有哪些优化办法

1.丢弃订单：最早期，量太大扛不住，直接前端随机 reject 一些，返回给抢单失败，简单粗暴，但是有效，比如10万人抢100个 iPhone，只要能提前预测有大概1万以上的人参与（通过资格确认、报名等方式收集信息），那么直接请求进来以后随机挡回去99%的流量都没有啥问题。

2.优化吞吐：中间有段时间，提前准备一大批机器，服务化、分库分表搞定后端性能，让前端业务可以加一定量的机器，然后搞稳定性，依赖关系，容量规划，做弹性，提升吞吐量。

3.异步队列：然后就是使用可堆积的消息队列或者内存消息队列了，如果抢单具有强顺序，那么先都进队列，然后拿前N（就是库存数）个出来平滑处理，剩下的所有都可以作为失败进行批处理了，甚至还可以做一个定长的队列，再往里写直接提示失败。队列把并发变成串行，从而去掉了锁。

技术上有哪些优化办法

4.内存分配：一些具体的业务，也会考虑预热，提前在每个机器节点内存分配好库存数量，然后直接在内存里处理自己的库存数即可，这样可能也会在极端情况下会不精确。

5.拆分扩展：针对不同类型、不同商家、不同来源的商品，部署不同的前端促销集群，这样就把压力分散开了。具体到每个商家，其实量就不大了，双十一销售第一名的商家，并发也不是特别高。

6.服务降级：越重要的抢单，大家越关心自己有没有抢到，而不是特别在意订单立即处理完，也就是说，下单占到位置比处理完成订单要更有价值。比如12306春运抢票，只要告诉用户你抢到了票，但是预计1个小时后订单才会处理完，用户有这个明确预期，就可以了，用户不会立马使用这张票，也不会在意1分钟内处理完还是1小时处理完。

需要注意的是其中部分模式会导致销售不足或者超卖，销售不足可以从抢购里加一些名单补发，也可以加一轮秒杀。超卖比较麻烦，所以一般会多备一点货，比如抢100个 iPhone，提前准备105个之类的，也会证明在实际操作里非常有价值。

系统设计上的改进和优化

将营销促销活动，从交易业务系统里剥离出来，成为独立的系统。

促销活动：满减，满赠，折扣等。

业务因素：用户范围，商品范围，使用时限，发放形式，还是账务相关、预算与核销等。

其他：

- 优惠券
- 红包
- 积分

秒杀业务本质上是一个特殊的折扣活动。

2. 具体要做哪些事情

做系统架构设计的步骤

1、分析现状

1.1 明确具体需求。特别是要挖掘和明确非功能性需求。

1.2 分析可行性。明确可行性与相关技术指标。

2、寻找路径

2.1 实现整体方案设计。

2.2 完成 POC 验证和关键技术选型。

3、确定方案

3.1 根据分析设计出最终方案，并与各方达成一致。

3.2 完善方案相关设计图和文档，成为项目研发的蓝图。

做系统架构设计的步骤

1、分析现状

产出：新需求文档，系统当前现状（包括业务和技术指标）的相关文档和设计图，可行性分析文档。

2、寻找路径

产出：设计方案初稿，关键问题分析，关键技术选型报告，POC 验证的场景设计文档和 DEMO，测试/压测结果等。

3、确定方案

产出：架构设计方案和设计图终稿，组织会议同步和宣贯。

3. 功能性与非功能性需求

功能性需求

商家：上架秒杀商品和库存，定义活动规则。

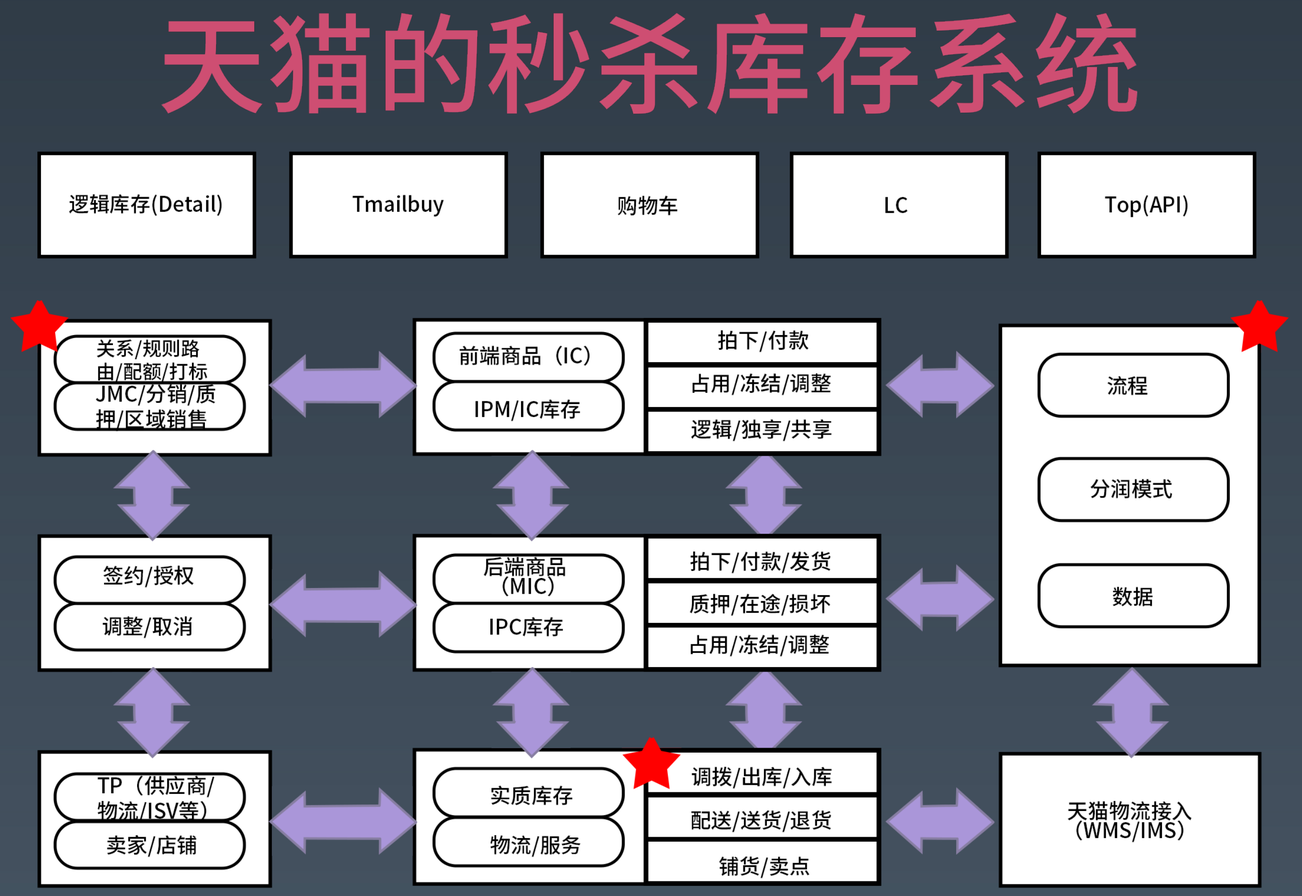
平台：(自营的话，不需要)

用户：满足条件，参与秒杀，抢购商品。

购买后的流程，与普通活动流程相同。

功能性需求

网上找了一个天猫的秒杀库存系统。



非功能性需求

根据业务指标，估算并发指标，以此来反推非功能性需求。

但是实际情况往往是，压力到底多大，不知道怎么办？

1、根据线上压测，现有系统指标的测量和计算
2、参考业内水平，设定并发流量的倍数
3、建立基线

注意事项

脱离场景谈性能，都是耍流氓

需要注意：不同互联网电商发展阶段，不同的现实条件下，对于秒杀系统的实际需求和解决办法，是有非常大的差异的。

PS：面试与实战的区别？

4. 如何编写设计文档

系统设计文档

传统软件开发一般有概要设计和详细设计。

互联网相关系统不会这么复杂，关键在于描述清楚我们的系统。

回忆一下，我们上一节讲的，什么方法可以达到这一点。

系统设计文档

1. 需求分析

2. 整体设计

3. 系统架构图

- 业务架构图
- 技术架构图
- 数据架构图
- 部署架构图

扩展到整个研发项目应该使用的文档结构

非常详细的研发文档结构。

PAGE TREE

- 1.平台整体业务与技术规划
  - 1.1.平台业务整体规划
  - 1.2.平台整体技术规划
- 2.项目管理与日常规范制度
  - 2.1.项目管理制度
  - 2.2.项目标准规范
  - 2.3.项目文档模版
  - 2.4.项目组人力资源池
  - 2.5.项目计划排期
  - 2.6.项目进度跟踪
  - 2.8.项目问题跟踪
  - 2.9.项目会议纪要
  - 2.10.项目结项文档
  - 2.11.其他日常工作
- 3.产品需求分析与原型设计
  - 3.1.需求分析文档
  - 3.2.产品原型设计
- 4.数据模型设计与日常管理
  - 4.1.数据库设计与管理规范
  - 4.2.项目数据库设计
  - 4.3.数据库设计评审
  - 4.4.日常开发测试支持
- 5.技术选型调研与架构设计
  - 5.1.技术管理标准与规范
  - 5.2.基础技术选型调研
  - 5.3.技术评审会议纪要
  - 5.4.基础技术平台维护
  - 5.5.技术问题处理汇总
- 6.系统编码实现与接口定义
  - 6.1.系统关键业务实现评审
  - 6.2.系统关键业务接口定义
  - 6.3.跨系统业务接口定义
- 7.系统测试工作与质量保障
  - 7.1.整体测试方案与计划
  - 7.2.日常功能与回归测试
  - 7.3.接口与UI自动化测试
  - 7.4.单元测试与集成测试
  - 7.5.性能测试与分析调优
  - 7.6.安全测试与加固修复
  - 7.7.持续集成与项目质量
- 8.系统运维管理与部署上线
  - 8.1.系统运维规范制度
  - 8.2.数据库运维规范制度
  - 8.3.上线部署操作流程
  - 8.4.日常运维管理事项
- 9.技术资料汇总与交流培训
  - 9.1.技术培训安排
  - 9.2.技术资料分类
  - 9.3.研发新手上路
- 10.其他工作事项与信息汇总

5. 如何考虑技术选型

技术选型的原则和方法

原则上，综合考虑采用相当成熟稳定的合适技术。

符合公司技术发展路线和选型规范（如果有）。

方法：基于关键场景编写 case，实现 demo，验证多种类似技术的各项指标。

是否要采用最新的技术

参考公司和团队的研发成熟度和技术能力水平，业界技术雷达，可以适度预研，不建议大规模采用不熟悉的新技术。

> *当你采用一个不熟悉的东西试图去解决已有问题，那么恭喜你，你现在有了2个问题*。

6. 其他相关技术要点

系统设计上的其他要考虑的要点

体系化的稳定性建设/混沌工程

系统设计上的其他要考虑的要点

架构团队与架构能力建设

系统设计上的其他要考虑的要点

营销活动工具的平台化建设

第 28 课总结回顾

架构具体要做什么

了解清楚具体需求

编写架构设计文档

其他相关设计要点

第 28 课作业实践

1、（必做）针对课上讲解的内容，自己动手设计一个高并发的秒杀系统，将架构图，设计文档等，提交到 GitHub。

2、（选做）针对自己工作的系统，或者自己思考的复杂场景，做系统性的架构设计。