如何设计一个秒杀系统

高性能、一致性、高可用。

架构原则：4要1不要

数据要尽量少：入参、出参尽量少；减少跟数据库的交互

请求要尽量少：较少客户端对js、css文件的请求数，将多个文件合并为一个返回

路径要尽量短：可以将远程调用改成JVM内部的方法调用

依赖尽量少：减少对次要系统的依赖，防止次要系统拖垮主流程

不要单点：应用无状态化，使用配置中心动态调整配置

动静分离：

用户请求的数据划分为动态数据和静态数据

动态数据跟用户请求的URL、浏览者、时间、地点以及Cookie的私密信息相关。

静态数据比仅仅是HTML页面，也可能是Java产生的页面。

静态数据优化：

使用缓存提高静态数据访问效率。

将静态数据放到离用户最近的地方。

动态数据加载：

ESI：服务端加载好所有动态数据一次性返回页面。服务器压力大，但响应快

CSI：客户端对多个动态数据分别请求，逐个加载。服务器压力小，但相应慢

架构优化：

秒杀系统单独部署服务、秒杀的不同模块单独部署集群集群、热点数据放缓存、增加秒杀答题防止秒杀器。

进一步的优化：

减少页面刷新、秒杀物品本地缓存

静态数据缓存：

静态HTML缓存在用户浏览器、CDN、服务器的Cache中。

服务端可以直接缓存http head和body，无需组装http协议。

服务器静态数据可以将缓存的从Java逻辑中抽出，直接放在服务器上，例如Nginx、Apache等，他们能解决大并发下请求静态文件。

消峰填谷：

消息队列

答题方式

分层过滤：CDN拦截读请求，前台系统通过Cache过滤无效请求，后台系统二次校验做好保护和限流，

热点数据：

静态热点数据：卖家上报和大数据计算