HomeWork 9- 秒杀

1. 设计一个秒杀系统，主要的挑战和问题有哪些？核心的架构方案或者思路有哪些?
    - 主要的挑战和问题：
    1. 对现有网站业务造成冲击
        1. 秒杀活动只是网站营销的一个附加活动，这个活动具有时间短，并发访问量大的特点，如果和网站原有应用部署在一起，必然会对现有业务造成冲击，稍有不慎可能导致 整个网站瘫痪。
    2. 高并发的应用，数据库负载
        1. 用户在秒杀开始前，通过不停刷新浏览器页面以保证不会错过秒杀，这些请求如果 按照一般的网站应用架构，访问应用服务器、连接数据库，会对应用服务器和数据库服 务器造成极大的负载压力。
    3. 突然增加的网络和服务器带宽
        1. 假设商品页面大小 200K(主要是商品图片大小)，那么需要的网络和服务器带宽是

           2G(200K×10,000)，这些网络带宽是因为秒杀活动新增的，超过网站平时使用的带宽。如何保证应用在这种复杂突发大流量情况下还能高效稳定运行，如何预防和面对突发问题?
    4. 直接下单
        1. 秒杀的游戏规则是到了秒杀时间才能开始对商品下单购买，在此时间点之前，只能 浏览商品信息，不能下单。而下单页面也是一个普通的 URL，如果得到这个 URL，不用 等到秒杀开始就可以下单了。
    5. 一致性问题
        1. 秒杀需要关注商品库存，有线的商品在同一时间被多个请求同时扣减，而且要保证准确性，显而易见是一个难题，如何做到既不多又不少?
    - 核心架构方案和思路
    1. 秒杀系统独立部署
        1. 为了避免因为秒杀活动的高并发访问而拖垮整个网站，使整个网站不必面对蜂拥而来的用户访问，可将秒杀系统独立部署;如果需要，还可以使用独立的域名，使其与网 站完全隔离，即使秒杀系统崩溃了，也不会对网站造成任何影响。
    2. 秒杀商品页面静态化
        1. 重新设计秒杀商品页面，不使用网站原来的商品详情页面，页面内容静态化:将商 品描述、商品参数、成交记录和用户评价全部写入一个静态页面，用户请求不需要经过 应用服务器的业务逻辑处理，也不需要访问数据库。所以秒杀商品服务不需要部署动态 的 Web 服务器和数据库服务器。
    3. 租借秒杀活动网络带宽
        1. 因为秒杀新增的网络带宽，必须和运营商重新购买或者租借。为了减轻网站服务器 的压力，需要将秒杀商品页面缓存在 CDN，同样需要和 CDN 服务商临时租借新增的出口 带宽。
    4. 动态生成随机下单页面 URL
        1. 为了避免用户直接访问下单页面 URL，需要将该 URL 动态化，即使秒杀系统的开发 者

也无法在秒杀开始前访问下单页面的 URL。办法是在下单页面 URL 加入由服务器端生成的随机数作为参数，在秒杀开始的时候才能得到。

5. 动静分离
    1. 动静分离三步走：1、数据拆分；2、静态缓存；3、数据整合。
6. 减库存的方式
    1. **下单减库存。**买家下单后，扣减商品库存。下单减库存是最简单的减库存方式，也是控制最为精确的一种
    2. **付款减库存。**买家下单后，并不立即扣减库存，而是等到付款后才真正扣减库存。但因为付款时才减库存，如果并发比较高，可能出现买家下单后付不了款的情况，因为商品已经被其他人买走了
    3. **预扣库存。**这种方式相对复杂一些，买家下单后，库存为其保留一定的时间（如 15 分钟），超过这段时间，库存自动释放，释放后其他买家可以购买
7. 流量削峰
    1. 对于秒杀的目标场景，最终能够抢到商品的人数是固定的，无论 100 人和 10000 人参加结果都是一样的，即有效请求额度是有限的。并发度越高，无效请求也就越多。但秒杀作为一种商业营销手段，活动开始之前是希望有更多的人来刷页面，只是真正开始后，秒杀请求不是越多越好。因此系统可以设计一些规则，人为的延缓秒杀请求，甚至可以过滤掉一些无效请求。常见的手段包括
        - 答题
            - 通过提升购买的复杂度，达到两个目的：
                - **防止作弊。**早期秒杀器比较猖獗，存在恶意买家或竞争对手使用秒杀器扫货的情况，商家没有达到营销的目的，所以增加答题来进行限制
                - **延缓请求。**零点流量的起效时间是毫秒级的，答题可以人为拉长峰值下单的时长，由之前的 <1s 延长到 <10s。这个时间对于服务端非常重要，会大大减轻高峰期并发压力；另外，由于请求具有先后顺序，答题后置的请求到来时可能已经没有库存了，因此根本无法下单，此阶段落到数据层真正的写也就非常有限了
        - 排队
            - 最为常见的削峰方案是使用消息队列，通过把同步的直接调用转换成异步的间接推送缓冲瞬时流量。除了消息队列，类似的排队方案还有很多，例如：
                - 线程池加锁等待
                - 本地内存蓄洪等待
                - 本地文件序列化写，再顺序读
        - 过滤
            - 过滤的核心结构在于分层，通过在不同层次过滤掉无效请求，达到数据读写的精准触发。常见的过滤主要有以下几层：

              > **1、读限流：**对读请求做限流保护，将超出系统承载能力的请求过滤掉
              > **2、读缓存：**对读请求做数据缓存，将重复的请求过滤掉
              > **3、写限流：**对写请求做限流保护，将超出系统承载能力的请求过滤掉
              > **4、写校验：**对写请求做一致性校验，只保留最终的有效数据

              过滤的核心目的是通过减少无效请求的数据IO保障有效请求的IO性能。
8.