## 接口限流

**作用：**

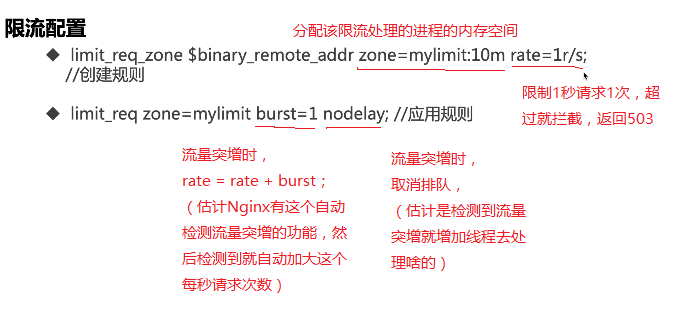
预防接口流量突然暴增，而造成服务器雪崩。

**实现：**

Nginx自带限流功能，修改配置即可。

Nginx限流方案有两种

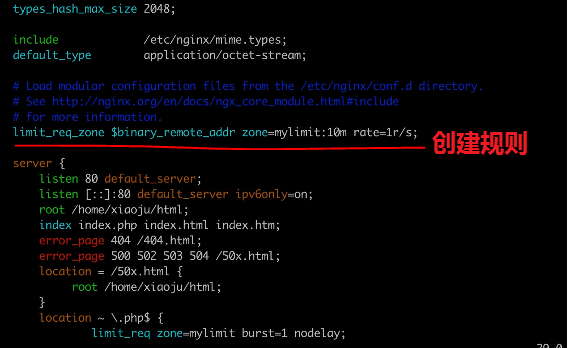




**DEMO：**

打开nginx.conf

修改，





## 限流算法

##### 令牌算法：

优点：可以处理突发流量，漏桶不行

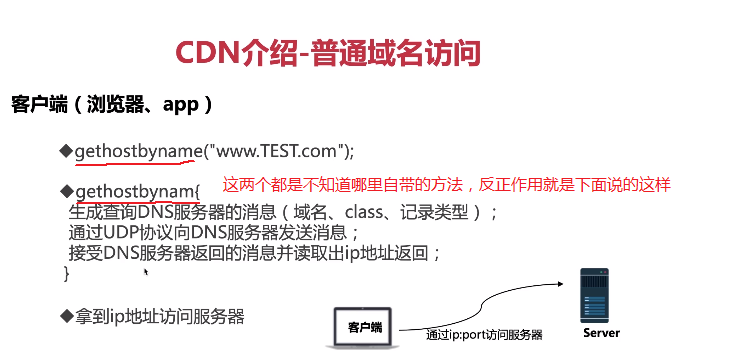
##### 漏桶算法：

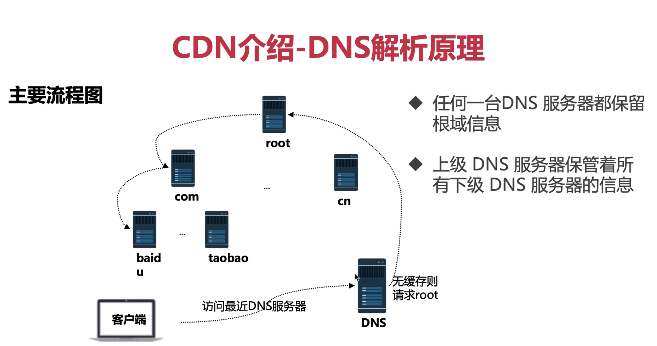
（注：感觉令牌跟漏桶的区别就是多了一个时间限制，令牌是每秒可以接受请求多少次 ，加入在高并发情况下，用户请求失败了，前端检测到这个令牌不足的标志就重新请求，可能就进去了。而漏桶需要慢慢排队，可能再请求几次也进不去。）

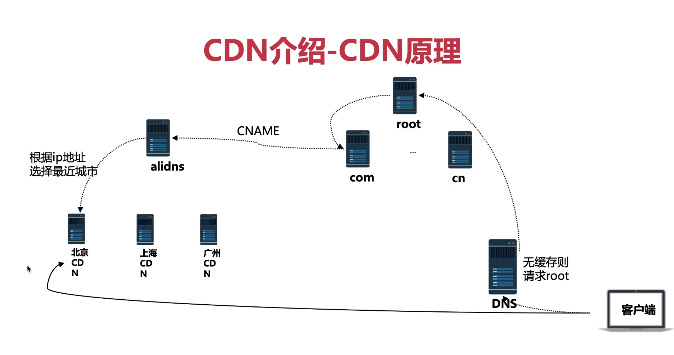
##### 计数：

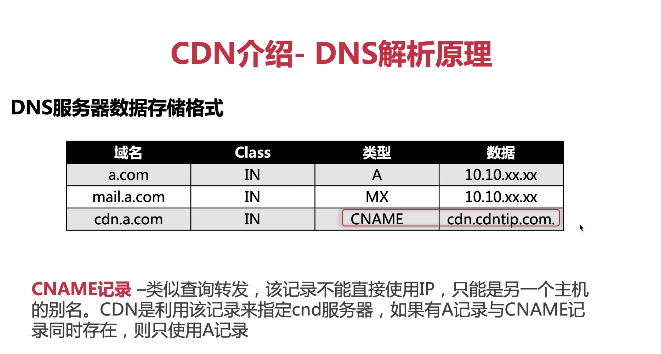
描述：上面两个都是在nginx配置中处理，这个是在代码里做处理，就是鸡肋的接到请求后记录一波ip什么的，看他规定时间内请求了多少次，超了就直接返回，不做业务处理

## CDN原理









配置了cdn的域名，是会解析到发起请求客户端最近的cdn服务器ip的。

所以微信开发的项目正常是不能配cdn的，因为微信有服务器ip认证。

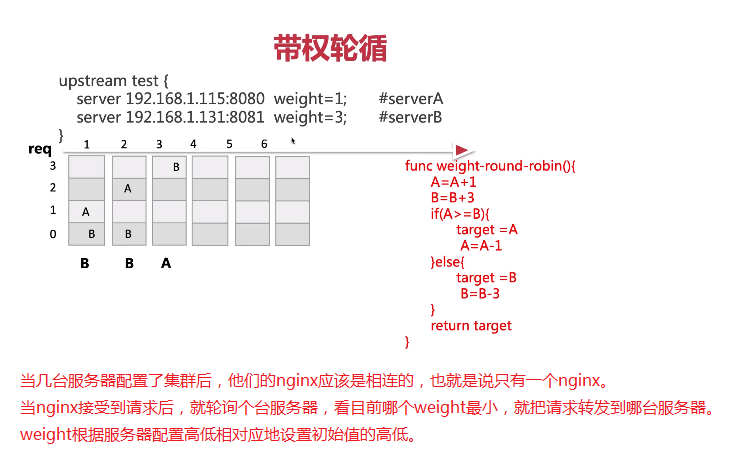
一般是用两个或以上的服务器去做，专门一个域名和服务器去做微信的部分，其他的做业务。

## 大型网站架构

##### Lvs四层负载均衡：

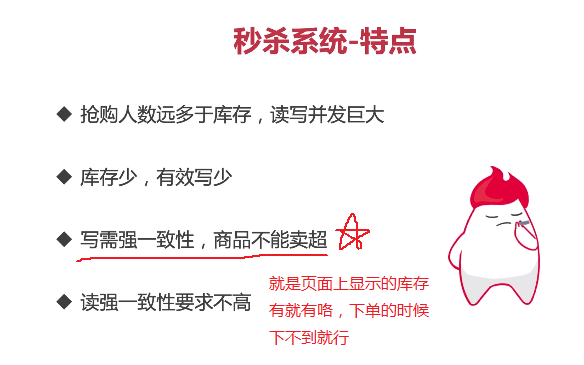
这一层就先不细究了

##### Nginx七层负载均衡：



## 秒杀系统难点分析与架构原则

##### 秒杀系统特点：



##### 架构原则：

**稳定性：**

高并发隔离，把高并发的接口放到专门一台服务器，别崩起来，全部崩了。

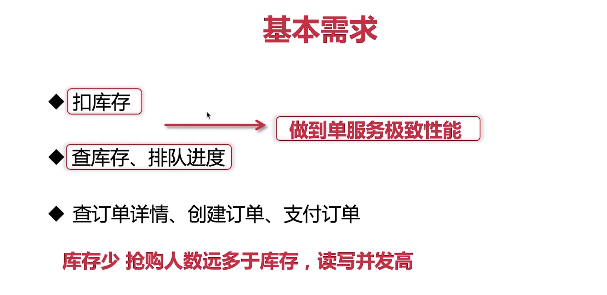
降级、限流，尽量避免崩了

**高性能：**

能放前端处理放前端，能在前端拦截就前端拦截

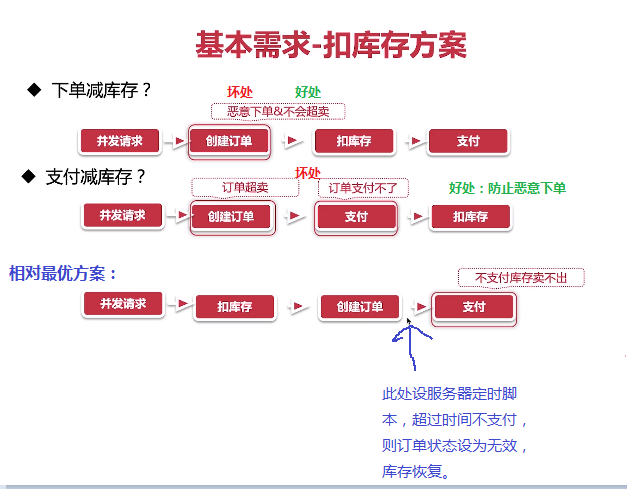
## 秒杀系统实现

##### 基本需求：



##### 基本方案：

**流程设计：**



**流程优化：**



**分布式：**

疑问：如何在高并发情况下保证多台服务器库存数据的一致性？

回答：可以在秒杀活动设定时就将库存平均分到每台机器的内存里。

弊端就是可能会因为负载均衡的分配或者机器故障有某些机器的库存没卖完......

##### 代码实现：

**代码流程：**

**代码展示：**