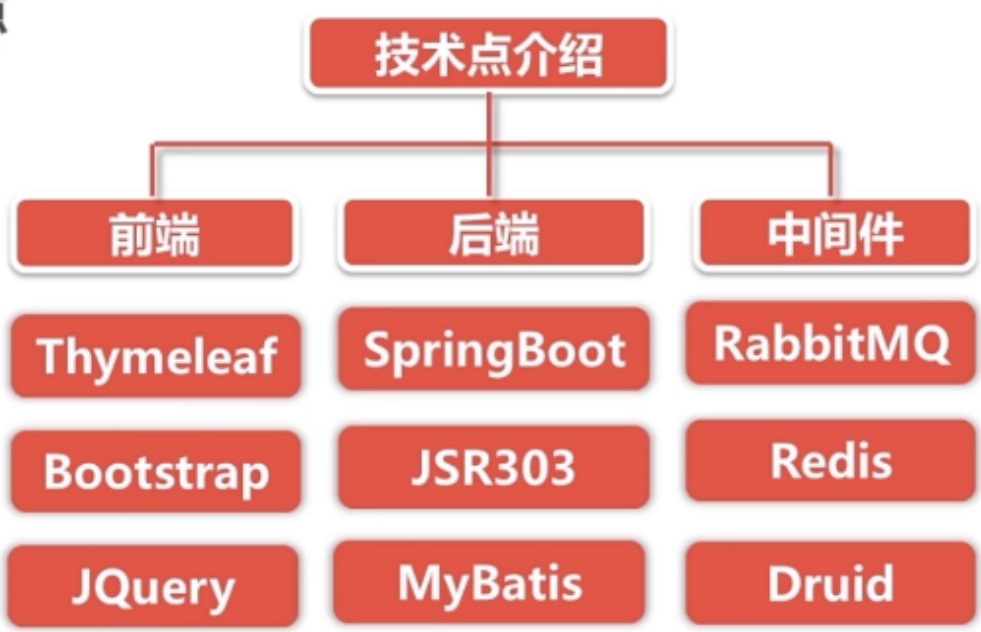
java 高并发-秒杀

目标：

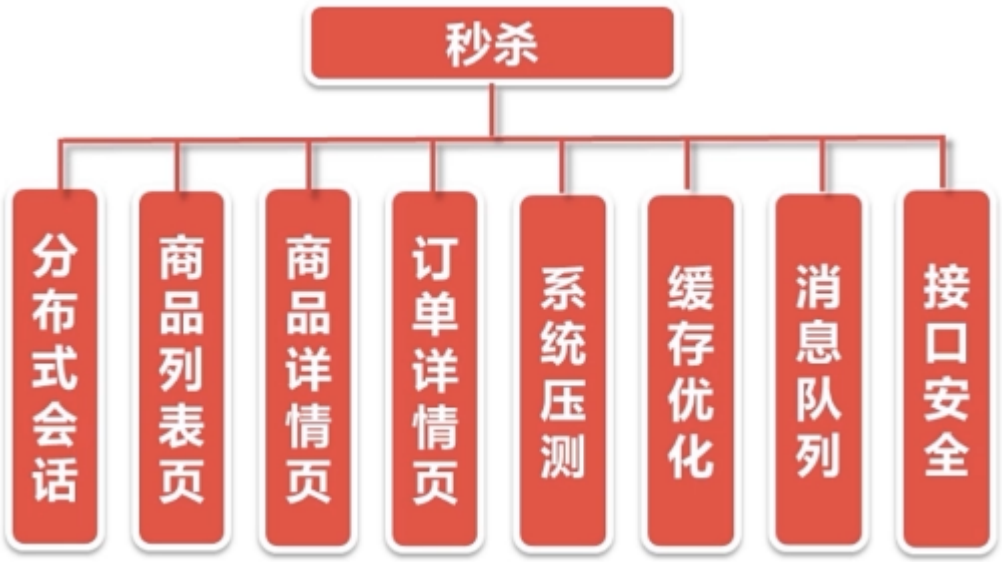
- 秒杀核心技术
    - 缓存
    - 异步化
- 不仅仅是秒杀
    - 应对大并发
        - 如何利用缓存
        - 如何使用异步
        - 应用扩展，分布式，负载均衡
        - 如何编写优雅的代码（封装）

技术点：JSR303 - validation 服务端验证框架

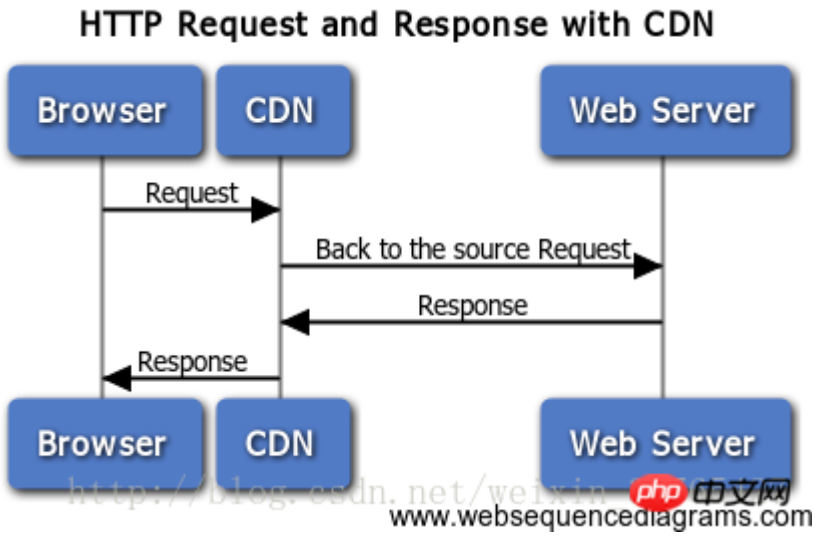
课程技术点



介绍：



缓存优化：缓存是一个很大的概念



客户端浏览器先检查是否有本地缓存是否过期，

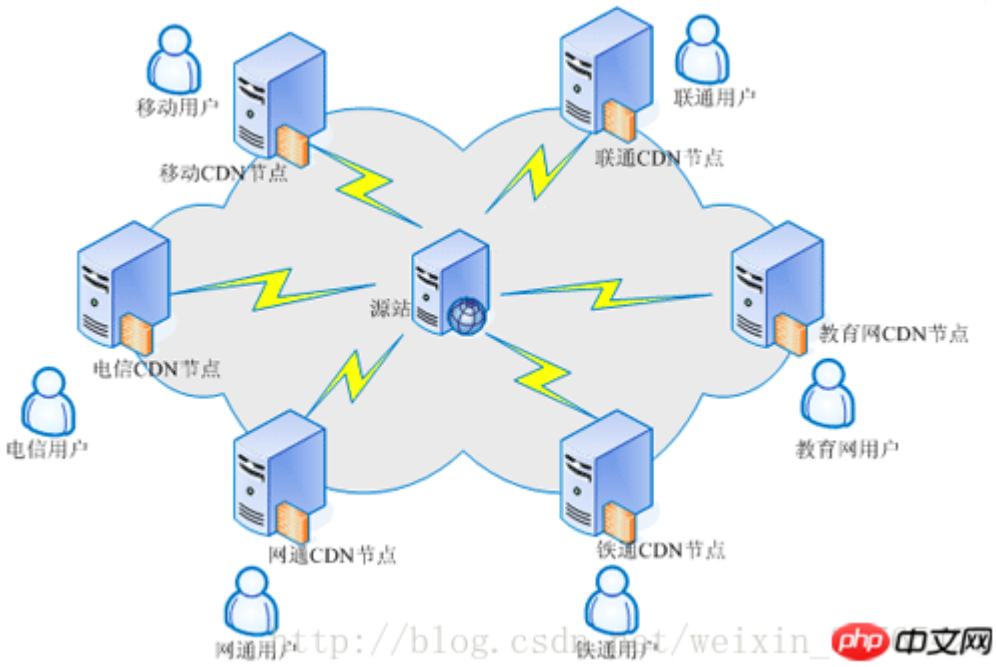
如果过期，则向CDN边缘节点发起请求，

CDN边缘节点会检测用户请求数据的缓存是否过期，

如果没有过期，则直接响应用户请求，此时一个完成http请求结束；

如果数据已经过期，那么CDN还需要向源站发出回源请求（back to the source request），来拉取最新的数据。

CDN的典型拓扑图如下：



CDN的优势很明显：

（1）CDN节点解决了跨运营商和跨地域访问的问题，访问延时大大降低；

（2）大部分请求在CDN边缘节点完成，CDN起到了分流作用，减轻了源站的负载。

--》用户手机端/浏览器端缓存静态页面

--》CDN （火车票代售点

--》网关（nginx缓存）

--》应用服务器 （页面缓存-整个页面缓存到redis；url-缓存；对象缓存）

缓存是应对高并发非常有效的手段，但缓存的问题：数据的不一致，如何权衡？先写数据库还是先写缓存？

使用消息队列rabbit，异步下单；

12306下单 --》排队中 而非直接告诉成功还是失败

使用nginx对应用横向扩展；若没有前面的优化过程仅仅只是增加服务器，是没法支持高并发的，所有的请求还是直接透传到DB，因为整个并发的瓶颈是数据库，只有减轻了数据库的压力，扩展才有意义。

接口安全优化：程序最终是运行在互联网上，恶意用户，竞争对手等

1）保证活动的相对公平

2) 保证系统不宕机：防刷，限流，验证码 ，隐藏接口

2.1 防刷：https://mp.weixin.qq.com/s/k7h8Q1OMG2xcz7zwoI2giw

2.2 限流：https://www.toutiao.com/i6752687084063949325/?timestamp=1572341336&app=news_article&group_id=6752687084063949325&req_id=20191029172855010026079016111EC747

应用级限流

2.2.1RateLimiter控制的是速率 **令牌桶算法**

2.2.2 Samephore控制的是并发量；**信号量**

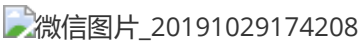
2.2.3 控制单位时间窗口内请求数

假设将应用部署到多台机器，应用级限流方式**只是单应用内的请求限流，不能进行全局限流。**

因此我们需要分布式限流和接入层限流来解决；

分布式限流

自定义注解+拦截器+Redis实现限流 (单体和分布式均适用，全局限流)



接入层限流

主要介绍nginx 限流，采用漏桶算法。

限制原理:可一句话概括为："根据客户端特征，限制其访问频率"，客户端特征主要指IP、UserAgent等。

使用IP比UserAgent更可靠，因为IP无法造假，UserAgent可随意伪造。

第一章 项目框架搭建

第二章 实现登录功能

第三章 实现秒杀功能

第四章 Jmeter压测

第五章 页面技术优化

第六章 接口优化


Redis预减库存减少数据库访问
内存标记减少Redis访问
RabbitMQ队列缓冲，异步下单，增强用户体验
RabbitMQ安装与Spring Boot集成
访问Nginx水平扩展
压测

第七章 安全优化


秒杀接口地址隐藏
数学公式验证码
接口防刷