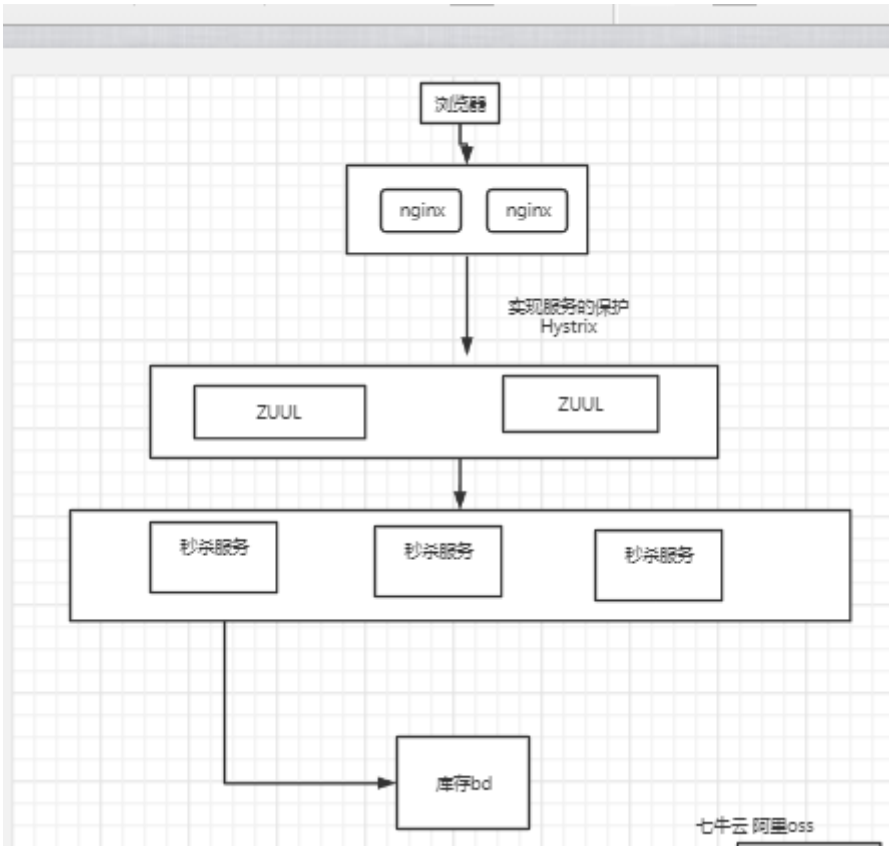
没有实战，等找到工作后再实战　基于理论

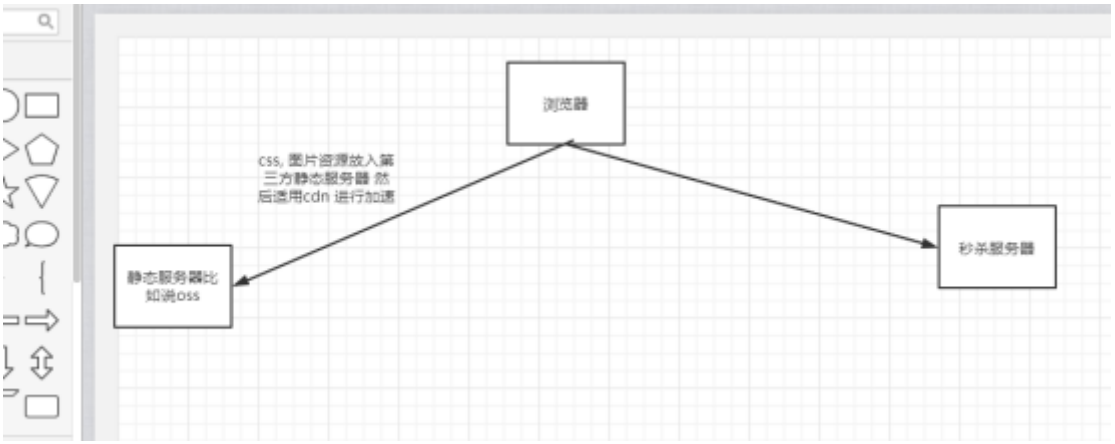
首先微服务架构肯定是浏览器第一步访问的是nginx 然后再访问网关然后再路由到我我们的微服务也就是说秒杀的微服务上[此时秒杀微服务需要单独部署



秒杀分前端和后端 的设计

在前端的设计：[动静分离：比方说前端的css ，图片，这样就可以让更多的用户来访问我的秒杀服务]

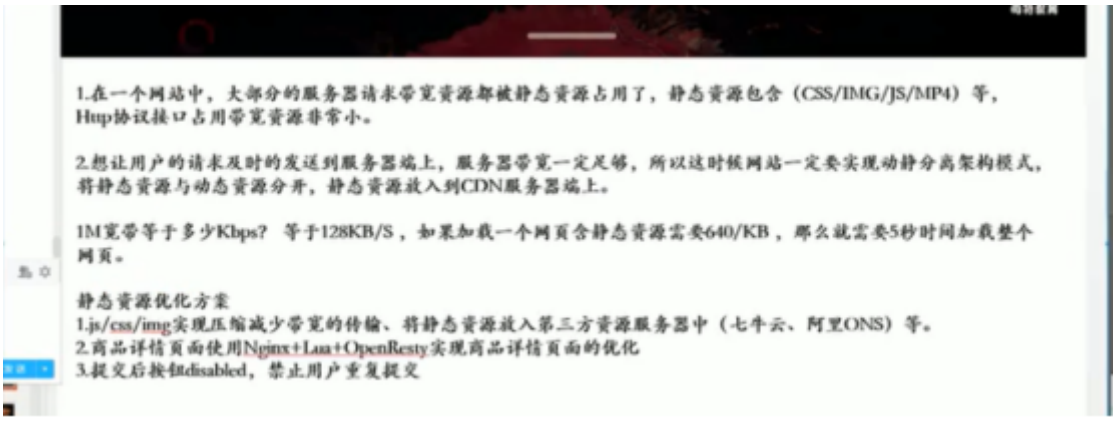
第一步 服务器是有带宽 服务器带宽指在特定时间段从或向网站/服务器传输的数据量;但是静态资源会占用一些带宽，所以此时第一步就是将静态资源和动态资源分开，将静态资源放到第三方服务器比方说阿里的**OSS**



使用**cdn** 内容分发， 遵循**Ip**就近原则 ；cdn把你的内容缓存到全国各地 再进行就近原则让最近的服务器进行访问 可以减少服务器的带宽传输

第二步： 对于商品详情页面一般不会发生变化，所以不会让好多人都去加载商品详情页此时可以用**nginx** 缓存页面,吧页面缓存到**nginx** 因为此时商品详情页很少变化{根据**url**地址实现缓存}

3. 提交后将按钮置灰，防止重复提交



----------->基于后端操作

1.>首先秒杀表结构

库存表

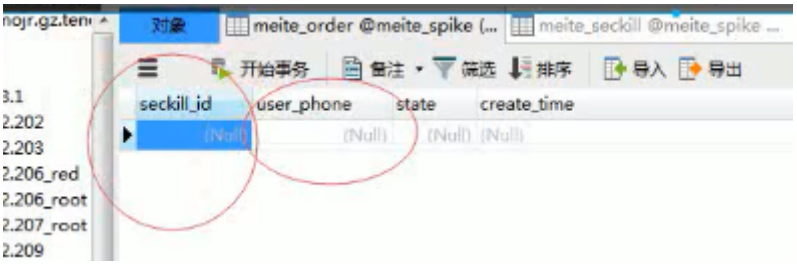
```
CREATE TABLE `meite_seckill` (
  `seckill_id` bigint(20) NOT NULL COMMENT '商品库存id',
  `name` varchar(120) CHARACTER SET utf8 NOT NULL COMMENT '商品名称',
  `inventory` int(11) NOT NULL COMMENT '库存数量',
  `start_time` datetime NOT NULL COMMENT '秒杀开启时间',
  `end_time` datetime NOT NULL COMMENT '秒杀结束时间',
  `create_time` datetime NOT NULL COMMENT '创建时间',
  `version` bigint(20) NOT NULL DEFAULT '0',
  PRIMARY KEY (`seckill_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='秒杀库存表';
```

库存表中数据：

商品库存Id 商品名称 库存量

meite_seckill seckil_id name inventory

订单表：



订单表中

订单表中数据：

// 商品Id 用户信息 状态 创建时间

1.meite_order seckil_id userPhone state time

---------->--------

2.>如何防止超卖问题

```
//------>秒杀接口的设计 伪代码
{保证唯一}当用户点击立即下单 会传用户信息 1.手机号码(或者UserId) 2.秒杀库存商品Id
public Object spike(String phone, Long seckileeId) {
 //1.参数验证 判断

 //2. 用户频率限制
 [因为你要对用户做一个频率限制 不可能一个用户每秒钟都会访问我接口 以后会在网关去做]
 //3.修改数据库对应的库存 减少库存[调用dao接口]
 update meite_seckill set inventory=inventory-1 where inventory>0 and seckil_id =101 // 直接dao修改库存表
 //4.添加秒杀订单 基于MQ异步形式
 return null;
}
```

```
//3.修改数据库对应的库存 减少库存[调用dao接口]
版本1
```

```
3  update meite_seckill set inventory=inventory-1 where inventory>0 and sec
kil_id =101 // 直接dao修改库存表
4  此时就会防止超卖 加个判断 inventory>0 因为在Mysql当中 每次在做更新数据库的时
候
5 有行锁的机制 所以不存在超卖的问题
6 ----------------------------------------------------------------------
--------》
7 版本2 基于版本号形式实现数据库乐观锁 {第二 是乐观锁控制 使用version // 先加个版
本号}
8  在库存表中加个版本号 version
9  分2部 第一步先查出来版本号
10  select version from meite_seckill //比方说查出来的是1
11  再根据版本号进行修改
12  update meite_seckill set inventory= inventory-1,version=version+1 from
meite_seckill where seckill_id='10001',and version="1"
13
14  当2个线程同时执行的时候 当第线程1和线程2同时查询出来的版本号是01 ，此时线程1就
会去根据版本号去修改库存， 然后也会修改版本号
15  线程2自然讲就是修改库存失败[也就是抢购失败]
16  ------------------------------------------------------------
```

```
1
2 第一 使用数据库自带的行锁机制进行实现 是悲观
3 {
4 1.200个请求 100个请求成功
5 }
6 第二 是乐观锁控制 使用version // 先加个版本号
7 {
8 1.查询版本号 2 版本号对应就更改库存 //可以少卖
9 }
10 //--------->场景2 的效率比场景1 好点 抢购失败
11  加个版本号 防止一下子全被抢购成功{控制} 还有分时段限流抢购 比如说12306 分段卖
票
12  12: 00-1:00
13  进行分段抢购比较好
```

```
1  //2. 用户频率限制[待研究]//--->这个以后可以放在网关中
2  基于redis实现用户行为实现频率限制 使用redis 的setnx
3
4 Boolern selfAbsert= stringRedisTemplate.opsForValue().setIfAbsent( phone
seckileeId,101)// 过期时间
5 if(!selfAbsert){
```

```
6
7   sout -->访问次数过多 10s后再试
8 }
```

//-------->问题1 秒杀抢购修改数据库如何减少数据库io操作

同时有10万个请求来实现秒杀， 此时商品的库存只有100个， 此时只需要修改库存100次 而不是10W次 也就是说有（10W-100）个秒杀失败

比较靠谱点的方案就是 基于Mq+库存令牌桶实现

方案实现流程 提前对应商品库存生成好对应的令牌

提前生成好100个令牌 也就是100个令牌 在10W 个请求 中 只要谁能够获取到令牌 谁就能够执行减库存操作

获取到秒杀令牌中，再使用Mq异步实现修改来秒杀去减库存

此时基于多线程异步提前生成一个token，放入redis中，传参 一个是秒杀商品的id,一个是生成多少个token

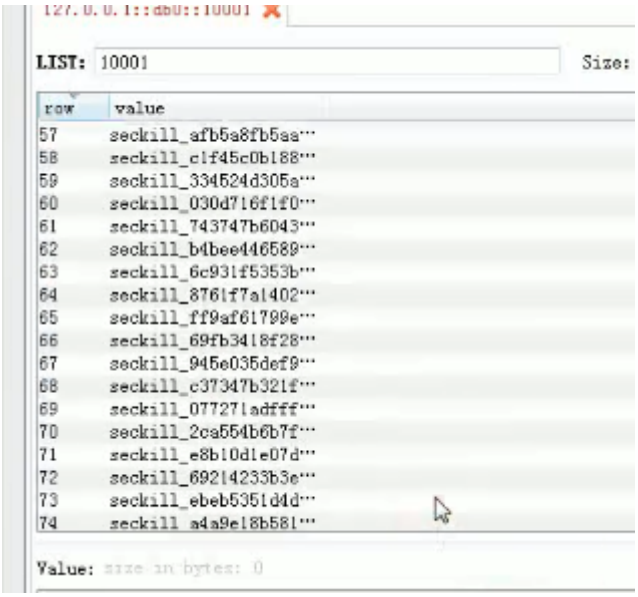
```
@Override
public BaseResponse<JSONObject> addSpikeToken(Long seckillId, Long tokenQuantity) {
    // 1.验证参数
    if (seckillId == null) {
        return setResultError("商品库存id不能为空!");
    }
    if (tokenQuantity == null) {
        return setResultError("token数量不能为空!");
    }
    SeckillEntity seckillEntity = seckillMapper.findBySeckillId(seckillId);
    if (seckillEntity == null) {
        return setResultError("商品信息不存在!");
    }
    // 2.使用多线程异步生产令牌
    createSeckillToken(seckillId, tokenQuantity);
    return setResultSuccess("令牌正在生成中.....");
}

@Async
private void createSeckillToken(Long seckillId, Long tokenQuantity) {
    generateToken.createListToken("seckill_", seckillId + "", tokenQuantity);
}
}
```

```java
50
51    public void createListToken(String keyPrefix, String redisKey, Long tokenQuantity) {
52        List<String> listToken = getListToken(keyPrefix, tokenQuantity);
53        redisUtil.setList(redisKey, listToken);
54    }
55
56    public List<String> getListToken(String keyPrefix, Long tokenQuantity) {
57        List<String> listToken = new ArrayList<>();
58        for (int i = 0; i < tokenQuantity; i++) {
59            String token = keyPrefix + UUID.randomUUID().toString().replace("-", "");
60            listToken.add(token);
61        }
62        return listToken;
63
64    }
```

Open Declaration
Open Implementation

大致就是这个样子



这样的话代码就是

```java
@Transactional
public BaseResponse<JSONObject> spike(String phone, Long seckillId) {
    // 1.参数验证
    if (StringUtils.isEmpty(phone)) {
        return setResultError("手机号码不能为空!");
    }
    if (seckillId == null) {
        return setResultError("商品库存id不能为空!");
    }
    // 2.从redis从获取对应的秒杀token
    String seckillToken = generateToken.getListKeyToken(seckillId + "");
    if (StringUtils.isEmpty(seckillToken)) {
        log.info(">>>seckillId:{}, 亲，该秒杀已经售空，请下次再来!", seckillId);
        return setResultError("亲，该秒杀已经售空，请下次再来!");
    }

    // 3.获取到秒杀token之后，异步放入mq中实现修改商品的库存
    sendSeckillMsg(seckillId, phone);
    return setResultSuccess("正在排队中.....");
}

/**
 * 获取到秒杀token之后，异步放入mq中实现修改商品的库存
 */
@Async
private void sendSeckillMsg(Long seckillId, String phone) {
    JSONObject jsonObject = new JSONObject();
    jsonObject.put("seckillId", seckillId);
    jsonObject.put("phone", phone);
    spikeCommodityProducer.send(jsonObject);
}

// 结业项目中采用rabbitmq实现秒杀
/**
 * 面试官 都喜欢问 你们项目中在那些地方使用到多线程
```

当高并发请求到接口的时候就会从redis中拿token 谁能拿到token[拿到token会有将这个token删除] 就把这个token放入到mq中 然后通过Mq异步实现秒杀扣库存 ，

Mq的监听者代码

```java
private SeckillMapper seckillMapper;
@Autowired
private OrderMapper orderMapper;

@RabbitListener(queues = "modify_inventory_queue")
@Transactional
public void process(Message message, @Headers Map<String, Object> headers, Channel channel) throws IOException {
    String messageId = message.getMessageProperties().getMessageId();
    String msg = new String(message.getBody(), "UTF-8");
    log.info(">>>messageId:{},msg:{}", messageId, msg);
    JSONObject jsonObject = JSONObject.parseObject(msg);
    // 1.获取秒杀id
    Long seckillId = jsonObject.getLong("seckillId");
    SeckillEntity seckillEntity = seckillMapper.findBySeckillId(seckillId);
    if (seckillEntity == null) {
        log.warn("seckillId:{},商品信息不存在!", seckillId);
        return;
    }
    Long version = seckillEntity.getVersion();
    int inventoryDeduction = seckillMapper.inventoryDeduction(seckillId, version);
    if (!toDaoResult(inventoryDeduction)) {
        log.info(">>>seckillId:{}修改库存失败>>>eduction返回为{} 秒杀失败! ", seckillId, inventoryDeduction);
        return;
    }
    // 2.添加秒杀订单
    OrderEntity orderEntity = new OrderEntity();
    String phone = jsonObject.getString("phone");
    orderEntity.setUserPhone(phone);
    orderEntity.setSeckillId(seckillId);
    orderEntity.setState(1l);
    int insertOrder = orderMapper.insertOrder(orderEntity);
    if (!toDaoResult(insertOrder)) {
        return;
    }
    log.info(">>>修改库存成功seckillId:{}>>>>inventoryDeduction返回为{} 秒杀成功", seckillId, inventoryDeduction);
```

呢么如果采用mq实现秒杀抢购，呢么秒杀接口会立马拿到秒杀结果么？

不会他会返回一个正在排队中，因为MQ 是一个异步的 帮我们解决流量削锋问题

想下我们在12306抢票时，也是啊返回结果为正在出票中....[等待10秒钟]// 此时会出单成功 或者是出单失败 对吧 一样的道理

呢么我怎么知道是秒杀成功或者说秒杀失败了

前端调用秒杀接口，如果秒杀成功的话 返回正在排队中

前端写一个定时器 使用秒杀token查询{用户信息 phone 和 秒杀商品id}是否秒杀成功

总结下

加入此时是100个库存 我们可以生成100个秒杀token放入redis中

redis什么样的数据类型　　redis key商品的库存id value是一个集合List

第一步.给对应商品库存生成令牌

第二步.秒杀抢购访问接口 时

{

1. 从redis中获取token 然后删除对应的token

//------>然后前端会写一个定时器来调用接口就是使用Phone和seckiiId 来查询秒杀状态

}

---------->大致就是这么个流程

我们可以通过生产者消息确认机制　来保证我们的消费一定确认消费了

还有手动消费

```
1  秒杀抢购：
```

```
[加载页面]1.前端：1.使用动静分离，将静态资源存放在第三方文件服务器中实现cdn加速， 减轻秒杀抢购的带宽
 2.将用户点击秒杀后 应该将按钮disabled，防止重复提交
 3.使用nginx+lua 缓存商品详情页实现静态化页面[并发大 但是不容易修改的]
 4.使用复杂的机器验证码防止机器模拟
 5.使用定时器 根据用户信息查询秒杀结果
```

```
 2.网关：1.限流 hystrix nginx redis 实现限流 算法令牌桶+漏斗算法 对用户的秒杀请求实现限流和服务保护，
  2.用户的黑名单和白名单拦截 防止恶意进行秒杀[吧他的 ip进行黑名单 呢么他此时就不会访问到我们秒杀接口]
```

```
3.秒杀接口：
 1.服务的降级隔离和熔断
 2.从redis当中获取秒杀令牌桶[能够获取到令牌就可以秒杀成功 ， 否则秒杀失败]
 3.异步使用Mq来执行修改库存操作[正在]
 4.提供一个根据 用户的信息查询秒杀结果接口[根据用户信息来查表]
---------------------------------------------------------->
高并发情况下 存在很多商品同时进行秒杀
 现在有100个商品同时进行秒杀抢购 每个商品库存100
 基于mq+库存令牌桶；10000 如何解决？ 数据库压力也大
 最靠谱的方案 分时段秒杀 不要同时去做秒杀
 比方12306买票的分时段卖票[分时段秒杀]
---------------------------------------------->
 其他点 分时段秒杀
```

分时段抢购 减少同时修改库存操作，我们知道限流操作添加到网关中,呢么如何限流

>1.用户限流频率设置 比方说 每秒钟 限制1个请求

>2.使用redis 限制用户的访问频率

java当中实现限流算法 令牌桶 漏桶 滑动窗口算法,

网关：

1. ratelimter、nginx、hystrix、redis 实现限流 令牌痛+漏铜算法 对用户秒杀请求实现限流和服务保护。

2. 用户黑名单和白名单拦截

秒杀接口：

1. 服务降级级、隔离、熔断

我们还可以做服务的降级 在接口中 利用Hystrix