|  |
| --- |
| 技术问题 |
| 该系统主要使广告主投放的广告和媒体方对接的平台。全面周到的技术咨询与支持；独特的防作弊技术；建立量身定做的网络广告营销体系。智能化的设计、完善的会员自助化管理功能、广告投放过程的全面监控、广告效果的实时 查验。·全程的客户服务跟踪，丰富的线上广告和营销经验支持，保证效果，省心省力。  使用大数据, 人工智能对用户行为特征,喜好进行分析,来完成媒体方对用户的广告投放.不断优化系统性能,提高广告转化率, 提供覆盖市面主流样式的合作，包括原生、开屏、插Banner等多种广告展现形式。同时，创投广告在样式创新的道路上走在行业最前沿。凭借丰富的信息流广告经验，独创原生广告样式，使广告与播放环境高度融合，即能形成一致的用户体验，同时给广告主带来丰厚的收入。  本项目主要用于广告投放，服务于广告主和媒体方，广告主在平上进行投放广告，我们平台进行数据处理，然后提供给媒体方，进行广告展示，并且对媒体方进行相应的计费。  业务需求会决定广告系统的一切；在需求明确的基础上，策略是广告系统自身的核心；对于广告而言，不可避免地会遇到定制化需求，产品经理要在需求和系统两个层面处理这种定制化；最后，同样重要的是广告系统的基础设施。  Eureka 客户端向 Eureka Server 注册的时候会提供一系列的元数据信息, 例如: 主机, 端口, 健康检查 url 等。  Eureka Server接受每个客户端发送的心跳信息, 如果在某个配置的超时时间内未接收到心跳信息, 实例会被从注册列表中移除。  Feign可以以声明的方式调用微服务。  SpringBoot的 Maven 插件, 能够以 Maven 的方式为应用提供 SpringBoot 的支持，可以将SpringBoot 应用打包为可执行的 jar 或 war 文件, 然后以通常的方式运行 SpringBoot 应用。  用户账户 -> 最高层级，用于定义广告主或代理商，只有有了用户才会有接下来的数据投放。  推广计划 -> 一类品牌或产品广告投放的规划，自身并不定义太多关于广告自身的信息，它会将信息打包下放到推广单元层级。  推广单元 -> 一个确定的广告投放策略，描述了投放广告的规则信息。  推广单元维度限制 -> 广告投放会有一些限制条件，例如只投放到北京、上海地区，对一些关键字进行投放等等。  广告创意 -> 展示给用户看到的数据，可以是图片、文本或者一段视频。  投放系统是比较简单的模块，其核心实现的功能就是对广告数据（各个表）进行增删改查，即能够让用户（广告主/代理商）对数据进行查看、上传、修改与删除。  创建 索 引 的 目 的 就 是 为 了 加 快 查 询 的 速 度 ， 如 果 没 有 索 引 ， M y S Q L 在 查 询 时 ， 只 能 从 第 一 条 记 录 开 始 然 后 读 完 整 个 表 找 到 匹配 的 行 。  Dubbox 致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。  分布式是需要按照功能点把系统拆分，拆分成独立的功能。单独为某一个节点添加服务器。需要系统之间配合才能完成整个业务逻辑。系统之间需要进行通信。  分布式优点：把模块拆分，使用接口通信，降低模块之间的耦合度。把项目拆分成若干个子项目，不同的团队负责不同的子项目。增加功能时只需要再增加一个子项目，调用其他系统的接口就可以。可以灵活的进行分布式部署。  分布式缺点：1、系统之间交互需要使用远程通信，接口开发增加工作量。2、各个模块有一些通用的业务逻辑无法共用。SOA是一种支持面向服务的架构样式。Zookeeper注册中心负责服务地址的注册与查找，相当于目录服务，服务提供者和消费者只在启动时与注册中心交互，注册中心不转发请求，压力较小。  当你建立一个 Maven 的项目时，Maven 会检查你的 pom.xml 文件，以确定哪些依赖下载。首先，Maven 将从本地资源库获得 Maven 的本地资源库依赖资源，如果没有找到，然后把它会从默认的 Maven 中央存储库  http://search.maven.org/  查找下载。如果没有找到，在Java.net或JBoss的储存库远程仓库中查找。如果没有找到，该过程将停止并将错误消息输出到 Maven 控制台。Pom.Xml配置中比较重要的参数是 groupId、artifactId、version，这三个属性确定唯一的一个项目。  登陆系统和检索系统的核心业务开发，首先广告主登陆的方式是手机号加动态验证码的方式，这里的话，为了短信接口的可用性，我们接入了两个短信接口，分别是阿里大于和容联。前台广告主点击发送短信之后，后台拿到手机号，生成一个随即的6位验证码，然后调用短信接口发送短信，如果发送成功，则将验证码存到redis，并设置过期时间，过期时间为两分钟，然后前台会根据手机接收到的短信输入验证码，我们进行从redis取出并进行判断，如果没有过期并且一致，就说明登陆成功，我们将广告主的IP+手机号+用户名+时间添加到登陆日志表里，并且会查看数据库里有没有这个手机号的用户，如果有，则代表是登陆，如果没有，则添加广告主信息到数据库，然后根据广告主的手机号生成token，并且设置有效期，一般都是7200S,然后返回给前台，前台每次向后台发送请求都会经过zuul网关，然后zuul网关进行拦截并且验证token，如果token有效并且正确，则验证通过，开始转发请求，进行处理业务。如果token为null或者是伪造的，则返回相应的错误信息，并且不会进行转发请求，这是登陆所涉及的逻辑，然后我负责的另一个模块就是检索系统，主要任务就是向媒体方提供数据，也就是向曝光系统提供数据，并且作为消费者监听Kafka，当有提供者，也就是投放系统将广告的数据发送到Kafka中时，检索系统会收到消息并且拿到数据，然后进行数据处理，存到Elasticsearch 搜索引擎和数据库里，当媒体方要拿数据的时候，会经过曝光系统调用检索系统，用的是SpringCloud的Feign组件，然后检索系统会从Elasticsearch 里面查询数据，返回给曝光系统，曝光系统进行数据处理，将合理的广告信息返回媒体方，进行展示广告，每个广告信息都会有多个回调函数，比如展示之后的回调函数，和点击之后的回调函数，会调用到计费系统进行相应的计费策略，计费策略有CPM、CPC、CPA、CPD等策略，  首先Kafka是一个高吞吐，高流量的消息中间件，适用于数据量非常大的情况下，Kafka会自带一个zookeeper ，用于kafka服务器的注册，包括生产者和消费者，元数据信息都会被注册到zookeeper里，用与监控和提供服务器IP地址。生产者为生产消息的一方，发送消息共有三种策略，①没有返回值，消息发送出去即可，这种策略不安全，一般不会用，②会当场得到返回值，从发送消息到接收返回值期间，生产者会一直等待，这会造成线程阻塞的情况，而且会浪费时间，但是会保证数据的安全，能保证消息肯定能发送到kafka,并且能被处理。③消息发送出去之后，会立即得到响应，可以执行之后的逻辑，但不会立即得到返回值，等到消费者处理完消息之后，会执行回调函数，并且有返回值，返回值和策略②的相同。但是不会造成阻塞，并且能保证消息的安全性。性能非常高，一般都会使用这个策略。此为生产者发送消息的三个策略。发送的消息要经过序列化处理，只需要设置序列化策略就行，取出的时候也是设置反序列化策略，不需要手动反序列化。消费者处理消息是以消费者组的形式存在的，一个分区不可以被一个消费者组中的多个消费者同时处理，只要在一个消费者组里面，每个消费者处理的分区都是不一样的，但是多个消费者组可以同时处理一个的分区。订阅的体现只是在消费者组与kafka之间，不是消费者与kafka之间。  我们先拿推广计划来说，在我们的程序启动时，将IndexFileLoader 类声明为组件，并使用@PostConstruct注解声明在启动时调用init方法执行。在我们这个方法中，通过字符缓冲输入流Files.newBufferedReader( Paths.get(fileName)读取日志文件，读取数据文件的每一行，并收集到list中去，泛型是String类型的，遍历list集合，将String转化为adPlanTable对象，并且传入一个枚举类ADD来执行添加方法。我们是通过创建一个线程安全的ConcurrentHashMap，将计划的ID作为主键，对象作为value存储到这个map中。在这里使用的是正排索引。那么当我们加载单元地域限制以及关键词时使用的是倒排索引。在倒排索引创建时，我们会去判断一下我们所传进来的关键词的单元ID是否存在，就是从unitMap中查询。然后将关键词限制的这个对象添加到keyWordUnitMap中，这个map是将关键词作为Key，推广单元的ID的set集合作为value，这是我们倒排索引的实现。  在检索系统前期，使用map存储索引。首先，索引分为两个，分别是全量索引和增量索引，全量索引就是每次项目启动的时候都会将日志文件里的数据全部查询出来，然后将索引存放到Map里面，增量索引就是监听日志文件，每当数据库有变化，都会出发一个event事件，然后我们根据事件的类型来操作索引map，就实现了增量索引，代码中，全量索是在indefileloder中的init方法上加了个@postconstruct标签，标签的意思就是程序启动的时候执行此方法，先用files的newBufferedReader方法，传入文件名，然后读取文件中的每一行数据，然后收集到list集合里，就是String类型的List，每一行都对应数据库里的一条数据，然后遍历将String转换为相应的实体对象，然后以id为键，对象为value，存到map集合中，map选用线程安全的ConcurrentHashMap。用的是正排索引，但是在加载关键字和地域限制的时候用的是倒排索引，在倒排索引的创建时，我们会判断一下我们所传进来的关键字的单元ID是否存在，如果存在就收集到set集合里面，这就是倒排索引，以上是用map做全两加载的流程。增量索引就是，会有一个BinlogRunner类，这个类实现了CommandLineRunner类，当继承这个类之后，就会在程序启动之后，执行这个类重写的run方法，在这个run方法里面，我们调用了一个方法，然后方法里创建了一个线程，并且是一个长链接，这个方法里面监听了mysql的binlog日志，每当数据库数据被改变，程序都会监听到，然后判断是什么操作，然后对map进行相对应的操作。这就是增量索引的流程。  问题：**如何对返回给媒体方的对应的创意数据进行筛选过滤？**  回答：首先根据流量类型对广告位类型和推广单元进行初始过滤,找到流量类型对应的推广单元,然后,使用地域,兴趣和关键词再次对推广单元进行筛选,同事也要考虑关键词之间的关系是and还是or,然后再次判断推广单元的对应的计划是都存在,如果存在根据单元和创意之间的对应关系找到相应的创意,最后根据广告位的信息对创意进行再次过滤,最后得到相应广告位对应的创意。  问题：**广告作弊行为可能存在的哪几种情况？**  回答：媒体为了获得更高的受益，广告平台为了获得更高的分成，广告主竞争对手为了消耗竞争对手的预算，降低竞争对手的广告效果，可能出现作弊的行为。  问题：**什么是流量劫持？**  回答：流量劫持，就是在无权投放广告的地方强行投放，或改变广告创意甚至落地页的内容。比如当去输入某一个网站地址的时候，后面会自动参数进行跳转；手机访问网页的时候，运营商会根据你的UA（UserAgent）来植入广告；还有就是常见的右下角广告（不是网站正常投放的广告）。一般来说，只有一些网络底层服务的提供商，如 DNS、CDN、电信运营商等，才有能力进行劫持。  问题：**关于eCPM？**  回答：全称expected cost per mile，意思是每进行一千次投放的预期收入。v具体计算方法为eCPM = CTR（点击率） \* bid（出价），CTR可以理解为用户点击该广告的概率，因此CTR与bid相乘就是这一次投放的预期收入，广告业务希望收入最大化，因此CTR的计算就是核心中的核心。出价高，但没人点击，广告平台没有收益；点击率高，但出价低，广告平台还是没有收益。最终应该按照广告的出价与CTR的乘积作为综合打分排序的依据，bid\*CTR。既然bid\*CTR是所有广告综合打分的依据，且出价bid又是广告主事先设定好的，那么实际上，广告排序问题的核心又转向了广告CTR的预测，CTR预测是推荐系统、广告系统、搜索系统里非常重要的一部分，是一个工程，算法，业务三方结合的问题。  问题：**在线广告的收费方式通常有几种？**  回答：CPM（Cost Per Mille）：按千次展示收费，广告被浏览1000人次产生一笔费用。按展示次数考核，故收入=产品用户数\*广告展示率  CPC（Cost Per Click）：按点击收费，广告被用户点击一次产生一笔费用。按点击考核，故收入=产品用户数\*广告展示率\*广告点击率  CPA（Cost Per Action）：按用户行为收费，比如用户点击广告之后进行了一次应用下载/网站注册后产生一笔费用。按行为（可以是注册、评论、付费等等）考核，故收入=产品用户数\*广告展示率\*广告点击率\*行为转化率  CPS（Cost Per Sale）：按销售额收费，对广告引导的商品销售额进行一定比例的提成。  问题：**说一下收入？**  回答：收入=广告流量\*转化率；广告流量=产品用户数\*广告展示率\*广告点击率 。那么最终得到：  收入=产品用户数\*广告展示率\*广告点击率\*转化率 或  收入=产品用户数\*（A广告展示率\*A广告点击率\*A转化率+B广告展示率\*B广告点击率\*B转化率+……）  问题：**如何提高流量？**  回答：产品用户数（一般称为日活：日活往往是一个产品的核心指标）：每天启动/打开产品的用户数  产品用户数=新增用户+留存用户+回流用户=新增用户+之前的新增用户\*留存率+流失用户\*召回率(增加新增用户、提升留存率、提升流失召回率)  广告展示率（展示/日活：广告展示率主要反映的是广告的前端设计逻辑）：广告展示用户数/产品用户数(增加广告位、提升广告的页面层级、提升展示频次)  广告点击率（广告点击率主要反映的是用户对广告的喜好程度）：广告点击用户数/广告展示用户数 （素材优化、增加效果、提升关联、迎合喜好、结合场景、结合行为）  转化率：不同广告业务的定义是不一样的。如果只是单一的流量变现广告，且考核指标为CPC的话，那么转化率≈1；如果是增值、会员业务的话，那么转化率=付费用户数/广告点击用户数  广告投放变化趋势：考核展示量——考核流量——考核有效流量。  问题：**ElasticSearch的存储数据过程？**  回答：客户端选择一个 node 发送请求过去，这个 node 就是 coordinating node（协调节点）。coordinating node 对 document 进行路由，将请求转发给对应的 node（有 primary shard）。实际的 node 上的 primary shard 处理请求，然后将数据同步到 replica node。coordinating node 如果发现 primary node 和所有 replica node 都搞定之后，就返回响应结果给客户端。Elasticsearch主要是在我的检索系统中用到，因为我的检索系统主要是对存储和提供数据的作用，当检索系统启动的时候，回去监听到kafka消息队列，每当投放系统往kafka中我所监听的主题发送消息的时候，我都会去将消息取出，并且将数据存到Elasticsearch中，然后当曝光系统要取数据的时候，我就相应的数据取出来返回，让曝光系统处理，Elasticsearch主要就是数据储存和提供的功能，因为Elasticsearch是一个性能高的搜索引擎，他可以自动帮我们创建索引，我们只需要将数据存进去即可，所以非常方便。  问题：**项目中为什么要使用kafka？**  回答：因为当用户主添加或更改数据时，我们不仅仅需要往数据库去添加，还需要往ElasticSearch中去添加，同时还有大数据的系统进行一些实时数据计算，日志采集。  问题：**广告项目内部微服务相互调用解决方案？**  回答：使用feign组件进行调用，我们会将微服务的元数据注册到eureka中，包括application.name 如果application.name一样，则表示是用一个服务的集群。通过feign进行调用，feign默认负载均衡方式是使用ribbon。  问题：**如何平衡用户体验？**  回答：有两种无效情况的流量即重复展示（对单一用户频繁，甚至用户已经产生了有效点击、转化行为）、无价值（对单一用户无需求）。我们可以对于长时间展示无操作，考虑更换广告内容、减少展示频次或不展示；对于已经产生有效行为，考虑更换广告内容或不展示；增加关闭按钮（若用户主动关闭X次，考虑减少展示频次或不展示）；直接用户反馈（考虑减少展示频次或不展示）；分析用户需求（通过机器学习方法建立用户模型，分析判断用户喜好，考虑针对性投放/更换广告）  问题：**和并发相关的方案？**  回答：同步和锁。同步更多指的是应用程序的层面，多个线程进来，只能一个一个的访问，java中指的是syncrinized关键字。 锁也有2个层面，一个是java中谈到的对象锁，用于[线程同步](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%90%8C%E6%AD%A5&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)；另外一个层面是数据库的锁；如果是分布式的系统，显然只能利用数据库端的锁来实现。  问题：**什么是乐观锁？**  回答：乐观锁意思是不锁定表的情况下，利用业务的控制来解决并发问题，这样即保证数据的并发可读性又保证保存数据的排他性，保证性能的同时解决了并发带来的脏数据问题。（乐观锁认为竞争不总是会发生，因此它不需要持有锁，将比较-替换这两个动作作为一个原子操作尝试去修改内存中的变量，如果失败则表示发生冲突，那么就应该有相应的重试逻辑。）（悲观锁认为竞争总是会发生，因此每次对某资源进行操作时，都会持有一个独占的锁，就像synchronized，不管三七二十一，直接上了锁就操作资源了。）  问题：**如何实现乐观锁？**  回答：前提：在现有表当中增加一个冗余字段，version版本号, long类型。 原理：1）只有当前版本号》=数据库表版本号，才能提交；  2）提交成功后，版本号version ++。 在ormapping增加一属性optimistic-lock="version"即可，样例片段如： <hibernate-mapping> <class name="com.insigma.stock.ABC" optimistic-lock="version" table="T\_Stock" schema="STOCK">  问题：**Object 对象的方法有哪些？分别有什么作用？该什么场景用？**  回答：clone方法：保护方法，实现对象的浅复制，只有实现了Cloneable接口才可以调用该方法，否则抛出CloneNotSupportedException异常。主要是JAVA里除了8种基本类型传参数是值传递，其他的类对象传参数都是引用传递，我们有时候不希望在方法里讲参数改变，这是就需要在类中复写clone方法。  getClass方法：final方法，获得运行时类型。  toString方法：该方法用得比较多，一般子类都有覆盖。  finalize方法：该方法用于释放资源。finalize方法在垃圾回收时一定会被执行，而如果在此之前显示执行的话，也就是说finalize会被执行两次以上，而在第一次资源已经被释放，那么在第二次释放资源时系统一定会报错，因此一般finalize方法的访问权限和父类保持一致，为protected。  equals方法：该方法是非常重要的一个方法。一般equals和==是不一样的，但是在Object中两者是一样的。子类一般都要重写这个方法。  hashCode方法：该方法用于哈希查找，可以减少在查找中使用equals的次数，重写了equals方法一般都要重写hashCode方法。这个方法在一些具有哈希功能的Collection中用到。一般必须满足obj1.equals(obj2)==true。可以推出obj1.hash- Code()==obj2.hashCode()，但是hashCode相等不一定就满足equals。不过为了提高效率，应该尽量使上面两个条件接近等价。如果不重写hashcode(),在HashSet中添加两个equals的对象，会将两个对象都加入进去。  wait方法：wait方法就是使当前线程等待该对象的锁，当前线程必须是该对象的拥有者，也就是具有该对象的锁。wait()方法一直等待，直到获得锁或者被中断。wait(long timeout)设定一个超时间隔，如果在规定时间内没有获得锁就返回。调用该方法后当前线程进入睡眠状态，直到以下事件发生。（1）其他线程调用了该对象的notify方法。（2）其他线程调用了该对象的notifyAll方法。（3）其他线程调用了interrupt中断该线程。（4）时间间隔到了。此时该线程就可以被调度了，如果是被中断的话就抛出一个InterruptedException异常。  notify方法：该方法唤醒在该对象上等待的某个线程。  notifyAll方法：该方法唤醒在该对象上等待的所有线程。  问题：**面向对象的特征有哪些方面？**  回答：有四个特征，即抽象（将同类对象的共同特征提取出来构造类）、继承（基于基类创建新类）、封装（将数据隐藏起来，对数据的访问只能通过特定接口）、多态性（不同子类型对象对相同消息作出不同响应）。  **问题：float f=3.4;是否正确？**  答:不正确。3.4是双精度数，将双精度型（double）赋值给浮点型（float）属于下转型（down-casting，也称为窄化）会造成精度损失，因此需要强制类型转换float f =(float)3.4; 或者写成float f =3.4F;  问题：**解释内存中的栈(stack)、堆(heap)和方法区(method area)的用法。**  回答：通常我们定义一个基本数据类型的变量，一个对象的引用，还有就是函数调用的现场保存都使用JVM中的栈空间；而通过new关键字和构造器创建的对象则放在堆空间，堆是垃圾收集器管理的主要区域，由于现在的垃圾收集器都采用分代收集算法，所以堆空间还可以细分为新生代和老生代，再具体一点可以分为Eden、Survivor（又可分为From Survivor和To Survivor）、Tenured；方法区和堆都是各个线程共享的内存区域，用于存储已经被JVM加载的类信息、常量、静态变量、JIT编译器编译后的代码等数据；程序中的字面量（literal）如直接书写的100、"hello"和常量都是放在常量池中，常量池是方法区的一部分。栈空间操作起来最快但是栈很小，通常大量的对象都是放在堆空间，栈和堆的大小都可以通过JVM的启动参数来进行调整，栈空间用光了会引发StackOverflowError，而堆和常量池空间不足则会引发OutOfMemoryError。  问题：**重载（Overload）和重写（Override）的区别。重载的方法能否根据返回类型进行区分？**  回答：方法的重载和重写都是实现多态的方式，区别在于前者实现的是编译时的多态性，而后者实现的是运行时的多态性。重载发生在一个类中，同名的方法如果有不同的参数列表（参数类型不同、参数个数不同或者二者都不同）则视为重载；重写发生在子类与父类之间，重写要求子类被重写方法与父类被重写方法有相同的返回类型，比父类被重写方法更好访问，不能比父类被重写方法声明更多的异常（里氏代换原则）。重载对返回类型没有特殊的要求。  问题：**描述一下JVM加载class文件的原理机制？**  回答：JVM中类的装载是由类加载器（ClassLoader）和它的子类来实现的，Java中的类加载器是一个重要的Java运行时系统组件，它负责在运行时查找和装入类文件中的类。由于Java的跨平台性，经过编译的Java源程序并不是一个可执行程序，而是一个或多个类文件。当Java程序需要使用某个类时，JVM会确保这个类已经被加载、连接（验证、准备和解析）和初始化。类的加载是指把类的.class文件中的数据读入到内存中，通常是创建一个字节数组读入.class文件，然后产生与所加载类对应的Class对象。加载完成后，Class对象还不完整，所以此时的类还不可用。当类被加载后就进入连接阶段，这一阶段包括验证、准备（为静态变量分配内存并设置默认的初始值）和解析（将符号引用替换为直接引用）三个步骤。最后JVM对类进行初始化，包括：1)如果类存在直接的父类并且这个类还没有被初始化，那么就先初始化父类；2)如果类中存在初始化语句，就依次执行这些初始化语句。类的加载是由类加载器完成的，类加载器包括：根加载器（BootStrap一般用本地代码实现，负责加载JVM基础核心类库rt.jar）、扩展加载器（Extension从java.ext.dirs系统属性所指定的目录中加载类库，它的父加载器是Bootstrap）、系统加载器（System又叫应用类加载器，其父类是Extension。它是应用最广泛的类加载器。它从环境变量classpath或者系统属性java.class.path所指定的目录中加载类，是用户自定义加载器的默认父加载器）和用户自定义类加载器（java.lang.ClassLoader的子类）。从Java 2（JDK 1.2）开始，类加载过程采取了父亲委托机制（PDM）。PDM更好的保证了Java平台的安全性，在该机制中，JVM自带的Bootstrap是根加载器，其他的加载器都有且仅有一个父类加载器。类的加载首先请求父类加载器加载，父类加载器无能为力时才由其子类加载器自行加载。JVM不会向Java程序提供对Bootstrap的引用。  问题：**抽象类（abstract class）和接口（interface）有什么异同？**  回答：抽象类和接口都不能够实例化，但可以定义抽象类和接口类型的引用。一个类如果继承了某个抽象类或者实现了某个接口都需要对其中的抽象方法全部进行实现，否则该类仍然需要被声明为抽象类。接口比抽象类更加抽象，因为抽象类中可以定义构造器，可以有抽象方法和具体方法，而接口中不能定义构造器而且其中的方法全部都是抽象方法。抽象类中的成员可以是private、默认、protected、public的，而接口中的成员全都是public的。抽象类中可以定义成员变量，而接口中定义的成员变量实际上都是常量。有抽象方法的类必须被声明为抽象类，而抽象类未必要有抽象方法。  问题：**Java 中会存在内存泄漏吗，请简单描述一下。**  回答：理论上Java因为有垃圾回收机制（GC）不会存在内存泄露问题（这也是Java被广泛使用于服务器端编程的一个重要原因）；然而在实际开发中，可能会存在无用但可达的对象，这些对象不能被GC回收，因此也会导致内存泄露的发生。例如Hibernate的Session（一级缓存）中的对象属于持久态，垃圾回收器是不会回收这些对象的，然而这些对象中可能存在无用的垃圾对象，如果不及时关闭（close）或清空（flush）一级缓存就可能导致内存泄露。  问题：**如何实现对象克隆？**  回答：有两种方式： 实现Cloneable接口并重写Object类中的clone()方法； 实现Serializable接口，通过对象的序列化和反序列化实现克隆，可以实现真正的深度克隆，  问题：**GC是什么？为什么要有GC？**  回答：GC是垃圾收集的意思，内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显式操作方法。Java程序员不用担心内存管理，因为垃圾收集器会自动进行管理。要请求垃圾收集，可以调用方法：System.gc() 或Runtime.getRuntime().gc() ，但JVM可以屏蔽掉显式的垃圾回收调用。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低优先级的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清除和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。在Java诞生初期，垃圾回收是Java最大的亮点之一，因为服务器端的编程需要有效的防止内存泄露问题，然而时过境迁，如今Java的垃圾回收机制已经成为被诟病的东西。移动智能终端用户通常觉得iOS的系统比Android系统有更好的用户体验，其中一个深层次的原因就在于Android系统中垃圾回收的不可预知性。垃圾回收机制有很多种，包括：分代复制垃圾回收、标记垃圾回收、增量垃圾回收等方式。标准的Java进程既有栈又有堆。栈保存了原始型局部变量，堆保存了要创建的对象。Java平台对堆内存回收和再利用的基本算法被称为标记和清除，但是Java对其进行了改进，采用“分代式垃圾收集”。这种方法会根据Java对象的生命周期将堆内存划分为不同的区域，在垃圾收集过程中，可能会将对象移动到不同区域：  伊甸园（Eden）：这是对象最初诞生的区域，并且对大多数对象来说，这里是它们唯一存在过的区域。  幸存者乐园（Survivor）：从伊甸园幸存下来的对象会被挪到这里。  终身颐养园（Tenured）：这是足够老的幸存对象的归宿。年轻代收集（Minor-GC）过程是不会触及这个地方的。当年轻代收集不能把对象放进终身颐养园时，就会触发一次完全收集（Major-GC），这里可能还会牵扯到压缩，以便为大对象腾出足够的空间。  问题：**比较一下Java和JavaSciprt。**  回答：JavaScript 与Java是两个公司开发的不同的两个产品。Java 是原Sun Microsystems公司推出的面向对象的程序设计语言，特别适合于互联网应用程序开发；而JavaScript是Netscape公司的产品，为了扩展Netscape浏览器的功能而开发的一种可以嵌入Web页面中运行的基于对象和事件驱动的解释性语言。JavaScript的前身是LiveScript；而Java的前身是Oak语言。 两种语言间的异同作如下比较：  基于对象和面向对象：Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象；JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object-Based）和事件驱动（Event-Driven）的编程语言，因而它本身提供了非常丰富的内部对象供设计人员使用。  解释和编译：Java的源代码在执行之前，必须经过编译。JavaScript是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行。（目前的浏览器几乎都使用了JIT（即时编译）技术来提升JavaScript的运行效率）。  强类型变量和弱类型变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中变量是弱类型的，甚至在使用变量前可以不作声明，JavaScript的解释器在运行时检查推断其数据类型。  代码格式不一样。  其实Java和JavaScript最重要的区别是一个是静态语言，一个是动态语言。目前的编程语言的发展趋势是函数式语言和动态语言。在Java中类（class）是一等公民，而JavaScript中函数（function）是一等公民，因此JavaScript支持函数式编程，可以使用Lambda函数和闭包（closure），当然Java 8也开始支持函数式编程，提供了对Lambda表达式以及函数式接口的支持。对于这类问题，在面试的时候最好还是用自己的语言回答会更加靠谱，不要背网上所谓的标准答案。  问题：**Java语言如何进行异常处理，关键字：throws、throw、try、catch、finally分别如何使用？**  回答：Java通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好的接口。在Java中，每个异常都是一个对象，它是Throwable类或其子类的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并可以对其进行处理。Java的异常处理是通过5个关键词来实现的：try、catch、throw、throws和finally。一般情况下是用try来执行一段程序，如果系统会抛出（throw）一个异常对象，可以通过它的类型来捕获（catch）它，或通过总是执行代码块（finally）来处理；try用来指定一块预防所有异常的程序；catch子句紧跟在try块后面，用来指定你想要捕获的异常的类型；throw语句用来明确地抛出一个异常；throws用来声明一个方法可能抛出的各种异常（当然声明异常时允许无病呻吟）；finally为确保一段代码不管发生什么异常状况都要被执行；try语句可以嵌套，每当遇到一个try语句，异常的结构就会被放入异常栈中，直到所有的try语句都完成。如果下一级的try语句没有对某种异常进行处理，异常栈就会执行出栈操作，直到遇到有处理这种异常的try语句或者最终将异常抛给JVM。  问题：**举例说明同步和异步。**  回答：如果系统中存在临界资源（资源数量少于竞争资源的线程数量的资源），例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就必须进行同步存取（数据库操作中的排他锁就是最好的例子）。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。事实上，所谓的同步就是指阻塞式操作，而异步就是非阻塞式操作。  问题：**描述下网页一个 Http 请求，到后端的整个请求过程。**  回答：app/web端发送域名到网关（有外网、内网），由nginx进行负载均衡和方向代理把请求下发给不同的网关，请求经过网关访问到微服务（只有内网，外网关闭）。  问题：**说一下JWT?**  回答：JSON Web Token（JWT）是目前最流行的跨域身份验证解决方案。JWT的原则是在用户登录，服务器经过身份验证之后，将生成一个JSON对象并将其发送回客户端，由客户端进行存储于Cookie或localStorage中（如果将它存储在Cookie中，就可以自动发送，但是不会跨域，因此一般是将它放入HTTP请求的Header Authorization字段中。Authorization: Bearer，当跨域时，也可以将JWT被放置于POST请求的数据主体中。）。此后客户端将在与服务器交互中都会带JWT。用于之后的每一次与服务器通信的身份安全校验。JWT对象为一个长字串，字符之间通过"."分隔符分为三个子串，各字串之间也没有换行符，每一个子串表示了一个功能块，总共有三个部分：JWT头、有效载荷和签名。JWT头部分是一个描述JWT元数据的JSON对象，签名使用的算法默认为HMAC SHA256，JWT令牌类型统一写为JWT，使用Base64 URL算法将JSON对象转换为字符串保存。有效载荷部分，是JWT的主体内容部分，也是一个JSON对象，包含需要传递的数据。 JWT指定七个默认字段供选择：iss发行人、exp到期时间、sub主题、aud用户、nbf在此之前不可用、iat发布时间、jti JWT ID用于标识该JWT，也可以自定义私有字段。默认情况下JWT是未加密的，任何人都可以解读其内容，因此不要构建隐私信息字段，存放保密信息，以防止信息泄露。签名哈希部分是对上面两部分数据签名，通过指定的算法生成哈希，以确保数据不会被篡改。在计算出签名哈希后，JWT头，有效载荷和签名哈希的三个部分组合成一个字符串，每个部分用"."分隔，就构成整个JWT对象。  问题：**怎样生成签名？**  回答：首先需要指定一个密码（secret）。该密码仅仅为保存在服务器中，并且不能向用户公开。然后使用标头中指定的签名算法（默认情况下为HMAC SHA256）根据以下公式生成签名。  HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload),secret)  问题：**JWT存在什么问题？**  回答：JWT默认不加密，但可以加密。生成原始令牌后，可以使用改令牌再次对其进行加密。当JWT未加密方法是，一些私密数据无法通过JWT传输。JWT不仅可用于认证，还可用于信息交换。善用JWT有助于减少服务器请求数据库的次数。JWT的最大缺点是服务器不保存会话状态，所以在使用期间不可能取消令牌或更改令牌的权限。也就是说，一旦JWT签发，在有效期内将会一直有效。JWT本身包含认证信息，因此一旦信息泄露，任何人都可以获得令牌的所有权限。为了减少盗用，JWT的有效期不宜设置太长。对于某些重要操作，用户在使用时应该每次都进行进行身份验证。为了减少盗用和窃取，JWT不建议使用HTTP协议来传输代码，而是使用加密的HTTPS协议进行传输。  问题：**说一下流程？**  回答：我个人主要负责登陆系统和检索系统的核心业务开发，首先广告主登陆的方式是手机号加动态验证码的方式，这里的话，为了短信接口的可用性，我们接入了两个短信接口，分别是阿里大于和容联。前台广告主点击发送短信之后，后台拿到手机号，生成一个随即的6位验证码，然后调用短信接口发送短信，如果发送成功，则将验证码存到redis，并设置过期时间，过期时间为两分钟，然后前台会根据手机接收到的短信输入验证码，我们进行从redis取出并进行判断，如果没有过期并且一致，就说明登陆成功，我们将广告主的IP+手机号+用户名+时间添加到登陆日志表里，并且会查看数据库里有没有这个手机号的用户，如果有，则代表是登陆，如果没有，则添加广告主信息到数据库，然后根据广告主的手机号生成token，并且设置有效期，一般都是7200S,然后返回给前台，前台每次向后台发送请求都会经过zuul网关，然后zuul网关进行拦截并且验证token，如果token有效并且正确，则验证通过，开始转发请求，进行处理业务，  问题：**在进行数据库编程时，连接池有什么作用？**  回答：由于创建连接和释放连接都有很大的开销（尤其是数据库服务器不在本地时，每次建立连接都需要进行TCP的三次握手，释放连接需要进行TCP四次握手，造成的开销是不可忽视的），为了提升系统访问数据库的性能，可以事先创建若干连接置于连接池中，需要时直接从连接池获取，使用结束时归还连接池而不必关闭连接，从而避免频繁创建和释放连接所造成的开销，这是典型的用空间换取时间的策略（浪费了空间存储连接，但节省了创建和释放连接的时间）。池化技术在Java开发中是很常见的，在使用线程时创建线程池的道理与此相同。基于Java的开源数据库连接池主要有：C3P0、Proxool、DBCP、BoneCP、Druid等。在计算机系统中时间和空间是不可调和的矛盾，理解这一点对设计满足性能要求的算法是至关重要的。大型网站性能优化的一个关键就是使用缓存，而缓存跟上面讲的连接池道理非常类似，也是使用空间换时间的策略。可以将热点数据置于缓存中，当用户查询这些数据时可以直接从缓存中得到，这无论如何也快过去数据库中查询。当然，缓存的置换策略等也会对系统性能产生重要影响，对于这个问题的讨论已经超出了这里要阐述的范围。  问题：**什么是DAO模式？**  回答：DAO（Data Access Object）顾名思义是一个为数据库或其他持久化机制提供了抽象接口的对象，在不暴露底层持久化方案实现细节的前提下提供了各种数据访问操作。在实际的开发中，应该将所有对数据源的访问操作进行抽象化后封装在一个公共API中。用程序设计语言来说，就是建立一个接口，接口中定义了此应用程序中将会用到的所有事务方法。在这个应用程序中，当需要和数据源进行交互的时候则使用这个接口，并且编写一个单独的类来实现这个接口，在逻辑上该类对应一个特定的数据存储。DAO模式实际上包含了两个模式，一是Data Accessor（数据访问器），二是Data Object（数据对象），前者要解决如何访问数据的问题，而后者要解决的是如何用对象封装数据。  问题：**说一下ACID事务？** 回答：ACID有四个特性，即原子性（所谓的原子性就是说，在整个事务中的所有操作，要么全部完成，要么全部不做，没有中间状态。对于事务在执行中发生错误，所有的操作都会被回滚，整个事务就像从没被执行过一样）、一致性（事务的执行必须保证系统的一致性，就拿转账为例，A有500元，B有300元，如果在一个事务里A成功转给B50元，那么不管并发多少，不管发生什么，只要事务执行成功了，那么最后A账户一定是450元，B账户一定是350元）、隔离性（所谓的隔离性就是说，事务与事务之间不会互相影响，一个事务的中间状态不会被其他事务感知）、持久性（所谓的持久性，就是说一单事务完成了，那么事务对数据所做的变更就完全保存在了数据库中，即使发生停电，系统宕机也是如此）这种特性简称刚性事物。问题：分布式事务产生的场景？ 回答：在分布式系统，都会垂直拆分数据库，就比如我做的点阅广告系统就有广告主数据库、物料数据库等业务组成，分为多个数据源，会产生分布式事物问题。 问题：**spring事务和分布式事务的区别是什么?** 回答：spring事务，本地事务；分布式事务是跨服务间的通讯（不同的数据库连接）。问题:**CPA理论？** 回答：CAP关于数据一致性(consistency)、服务可用性(availability)、分区容错性(partition-tolerance)。  数据一致性(consistency)：如果系统对一个写操作返回成功，那么之后的读请求都必须读到这个新数据；如果返回失败，那么所有读操作都不能读到这个数据，对调用者而言数据具有强一致性(strong consistency) (又叫原子性 atomic、线性一致性 linearizable consistency) 。服务可用性(availability)：所有读写请求在一定时间内得到响应，可终止、不会一直等待。分区容错性(partition-tolerance)：在网络分区的情况下，被分隔的节点仍能正常对外服务。  问题：**什么是XA接口？**  回答：XA是一个分布式事务协议，由Tuxedo提出。XA中大致分为两部分：事务管理器和本地资源管理器。其中本地资源管理器往往由数据库实现，比如Oracle、DB2这些商业数据库都实现了XA接口，而事务管理器作为全局的调度者，负责各个本地资源的提交和回滚。  问题：**什么是Jta？**  回答：作为java平台上事务规范JTA（Java Transaction API）也定义了对XA事务的支持，实际上，JTA是基于XA架构上建模的，在JTA 中，事务管理器抽象为javax.transaction.TransactionManager接口，并通过底层事务服务（即JTS）实现。像很多其他的java规范一样，JTA仅仅定义了接口，具体的实现则是由供应商(如J2EE厂商)负责提供，目前JTA的实现主要几种：J2EE容器所提供的JTA实现(JBoss)。独立的JTA实现:如JOTM，Atomikos.这些实现可以应用在那些不使用J2EE应用服务器的环境里用以提供分布事事务保证。如Tomcat,Jetty以及普通的java应用。  问题：**分布式事物解决方案？**  回答：分布式事物问题，在互联网公司比较常见，可以使用全局事物2pc（两段提交协议）、3pc(三段提交协议)，消息中间件、tcc、gts、提供回滚接口、分布式数据库。  问题：**索引的优缺点，什么时候使用索引，什么时候不能使用索引？**  回答：索引最大的好处是提高查询速度，缺点是更新数据时效率低，因为要同时更新索引对数据进行频繁查询进建立索引，如果要频繁更改数据不建议使用索引。  问题：**volatile类型变量提供什么保证?**  回答：volatile 主要有两方面的作用:1.避免指令重排2.可见性保证.例如，JVM 或者 JIT为了获得更好的性能会对语句重排序，但是 volatile 类型变量即使在没有同步块的情况下赋值也不会与其他语句重排序。 volatile 提供 happens-before 的保证，确保一个线程的修改能对其他线程是可见的。某些情况下，volatile 还能提供原子性，如读 64 位数据类型，像 long 和 double 都不是原子的(低32位和高32位)，但 volatile 类型的 double 和 long 就是原子的。  问题：**java中用到的线程调度算法是什么？**  回答：抢占式。一个线程用完CPU之后，操作系统会根据线程优先级、线程饥饿情况等数据算出一个总的优先级并分配下一个时间片给某个线程执行。  问题：**如果你提交任务时，线程池队列已满，这时会发生什么？**  回答：如果你使用的LinkedBlockingQueue，也就是无界队列的话，没关系，继续添加任务到阻塞队列中等待执行，因为LinkedBlockingQueue可以近乎认为是一个无穷大的队列，可以无限存放任务；如果你使用的是有界队列比方说ArrayBlockingQueue的话，任务首先会被添加到ArrayBlockingQueue中，ArrayBlockingQueue满了，则会使用拒绝策略RejectedExecutionHandler处理满了的任务，默认是AbortPolicy。  问题：**如何在两个线程间共享数据？**  回答：通过在线程之间共享对象就可以了，然后通过wait/notify/notifyAll、await/signal/signalAll进行唤起和等待，比方说阻塞队列BlockingQueue就是为线程之间共享数据而设计的。  问题：**阐述 Java 7 和 Java 8 的区别。**  回答：lambda 表达式，Java 8 版本引入的一个新特性。lambda 表达式允许你将功能当作方法参数或将代码当作数据。lambda 表达式还能让你以更简洁的方式表示只有一个方法的接口 (称为函数式接口) 的实例。方法引用，为已命名方法提供了易于阅读的 lambda 表达式。默认方法，支持将新功能添加到类库中的接口，并确保与基于这些接口的旧版本的代码的二进制兼容性。重复注解，支持在同一声明或类型上多次应用同一注解类型。类型注解，支持在任何使用类型的地方应用注解，而不仅限于声明。此特性与可插入型系统一起使用时，可增强对代码的类型检查。  问题：**代码的分支管理策略?**  回答：关于代码管理的分支和发布策略，目前主要有两种：一种是主干作为新功能开发主线，分支用作发布。另一种是分支用作新功能开发，主干作为稳定版的发布。  问题：**浅谈架构、框架、模式、模块、组件、插件、控件、中间件？**  回答：1、架构：软件架构，也成称为软件体系结构，简单地说就是一种设计方案，将用户的不同需求抽象成组件，且能够描述组件之间的通信和调用。软件架构会分析工程中的问题，针对问题设计解决方案，针对解决方案分析应具有的功能，针对功能设计软件系统的层次和模块及层次模块之间的逻辑交互关系，确定各个功能如何由这些逻辑实现。开发人员可以根据软件架构分析出来的层次和架构进行软件编写。   1. 框架：软件框架，是软件开发过程中提取软件的共性部分形成的体系结构。框架不是现成可用的应用系统，而是一个半成品，是一个提供了诸多服务，供开发人员进行二次开发，实现具体功能的程序实体。 2. 框架与架构的关系：框架不是架构，框架比架构更具体，更偏重于技术，而架构更偏重于设计；架构可以通过多种框架来实现。   3、模式：设计模式强调的是一个设计问题的解决方法，是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。  框架与设计模式的关系：设计模式研究的是对单一问题的设计思路和解决方法，一个模式可应用于不同的框架和被不同的程序语言所实现；而框架则是一个应用的体系结构，是一种或多种设计模式和代码的混合体。设计模式的思想可以在框架设计中进行应用。  架构与设计模式的关系：设计模式研究的是对单一问题的设计思路和解决方法，范畴比较小；而架构是高层次的针对体系结构的一种设计思路，范畴比较大。一个架构中可能会出现多个设计模式的思想。  4、模块：根据不同的标准，通常会说程序模块或功能模块，程序模块指的是一段能够实现某个目标的成员代码段，功能模块则用来说明一个功能所包含的系统行为。定义模块的原则是：高内聚和低耦合。  5、组件：组件是封装了一个或多个程序模块的实体。组件强调的是封装，利用接口进行交互。组件也称为构建。插件是组件的一个子类，就是将组件中具有某些特点的组件归为插件。  6、插件：插件属于组件，插件是组件的一个子类，就是将组件中具有某些特点的组件归为插件。插件是一种电脑程序，通过和应用程序的互动，来为应用程序增加一些特定的功能，仅靠插件是无法正常运行的，需要依赖于应用程序才能发挥自身功能。插件和应用程序之间通过接口进行交互。  7、控件：可视化的组件。  8、中间件：通过系统软件和应用软件之间连接的软件，以便于软件各部件之间的沟通。  **Apache Kafka**是一款流行的分布式数据流平台，它已经广泛地被诸如New Relic(数据智能平台)、Uber、Square(移动支付公司)等大型公司用来构建可扩展的、高吞吐量的、且高可靠的实时数据流系统。例如，在New Relic的生产环境中，Kafka群集每秒能够处理超过1500万条消息，而且其数据聚合率接近1 Tbps。可见，Kafka大幅简化了对于数据流的处理，因此它也获得了众多应用开发人员和数据管理专家的青睐。然而，在大型系统中Kafka的应用会比较复杂。如果您的consumers无法跟上数据流的话，各种消息往往在未被查看之前就已经消失掉了。同时，它在自动化数据保留方面的限制，高流量的发布+订阅(publish-subscribe，pub/sub)模式等，可能都会影响到您系统的性能。可以毫不夸张地说，如果那些存放着数据流的系统无法按需扩容、或稳定性不可靠的话，估计您经常会寝食难安了。为了减少上述复杂性，New Relic公司为Kafka集群在应对高吞吐量方面的20项最佳实践。我将从如下四个方面进行展开：Partitions(分区)、Consumers(消费者)、Producers(生产者)、Brokers(代理)  针对Partitions的最佳实践  • 了解分区的数据速率，以确保提供合适的数据保存空间。此处所谓“分区的数据速率”是指数据的生成速率。换言之，它是由“平均消息大小”乘以“每秒消息数”得出的。数据速率决定了在给定时间内，所能保证的数据保存空间的大小(以字节为单位)。如果您不知道数据速率的话，则无法正确地计算出满足基于给定时间跨度的数据，所需要保存的空间大小。同时，数据速率也能够标识出单个consumer在不产生延时的情况下，所需要支持的最低性能值。  • 除非您有其他架构上的需要，否则在写topic时请使用随机分区。在您进行大型操作时，各个分区在数据速率上的参差不齐是非常难以管理的。其原因来自于如下三个方面：  首先，“热”(有较高吞吐量)分区上的consumer势必会比同组中的其他consumer处理更多的消息，因此很可能会导致出现在处理上和网络上的瓶颈。  其次，那些为具有最高数据速率的分区，所配置的最大保留空间，会导致topic中其他分区的磁盘使用量也做相应地增长。  第三，根据分区的leader关系所实施的最佳均衡方案，比简单地将leader关系分散到所有broker上，要更为复杂。在同一topic中，“热”分区会“承载”10倍于其他分区的权重。  针对Consumers的最佳实践  如果consumers运行的是比Kafka 0.10还要旧的版本，那么请马上升级。在0.8.x 版中，consumer使用Apache ZooKeeper来协调consumer group，而许多已知的bug会导致其长期处于再均衡状态，或是直接导致再均衡算法的失败(我们称之为“再均衡风暴”)。因此在再均衡期间，一个或多个分区会被分配给同一组中的每个consumer。而在再均衡风暴中，分区的所有权会持续在各个consumers之间流转，这反而阻碍了任何一个consumer去真正获取分区的所有权。  调优consumer的套接字缓冲区(socket buffers)，以应对数据的高速流入。在Kafka的0.10.x版本中，参数receive.buffer.bytes的默认值为64 kB。而在Kafka的0.8.x版本中，参数socket.receive.buffer.bytes的默认值为100 kB。这两个默认值对于高吞吐量的环境而言都太小了，特别是如果broker和consumer之间的网络带宽延迟积(bandwidth-delay product)大于局域网(local area network，LAN)时。对于延迟为1毫秒或更多的高带宽的网络(如10 Gbps或更高)，请考虑将套接字缓冲区设置为8或16 MB。如果您的内存不足，也至少考虑设置为1 MB。当然，您也可以设置为-1，它会让底层操作系统根据网络的实际情况，去调整缓冲区的大小。但是，对于需要启动“热”分区的consumers来说，自动调整可能不会那么快。  设计具有高吞吐量的consumers，以便按需实施背压(back-pressure)。通常，我们应该保证系统只去处理其能力范围内的数据，而不要超负荷“消费”，进而导致进程中断“挂起”，或出现consume group的溢出。如果是在Java虚拟机(JVM)中运行，consumers应当使用固定大小的缓冲区(请参见Disruptor模式：http://lmax-exchange.github.io/disruptor/files/Disruptor-1.0.pdf)，而且最好是使用堆外内存(off-heap)。固定大小的缓冲区能够阻止consumer将过多的数据拉到堆栈上，以至于JVM花费掉其所有的时间去执行垃圾回收，进而无法履行其处理消息的本质工作。  在JVM上运行各种consumers时，请警惕垃圾回收对它们可能产生的影响。例如，长时间垃圾回收的停滞，可能导致ZooKeeper的会话被丢弃、或consumer group处于再均衡状态。对于broker来说也如此，如果垃圾回收停滞的时间太长，则会产生集群掉线的风险。  针对Producers的最佳实践  • 配置producer，以等待各种确认。籍此producer能够获知消息是否真正被发送到了broker的分区上。在Kafka的0.10.x版本上，其设置是acks;而在0.8.x版本上，则为request.required.acks。Kafka通过复制，来提供容错功能，因此单个节点的故障、或分区leader关系的更改不会影响到系统的可用性。如果您没有用acks来配置producer(或称“fire and forget”)的话，则消息可能会悄然丢失。  • 为各个producer配置retries。其默认值为3，当然是非常低的。不过，正确的设定值取决于您的应用程序，即：就那些对于数据丢失零容忍的应用而言，请考虑设置为Integer.MAX\_VALUE(有效且最大)。这样将能够应对broker的leader分区出现无法立刻响应produce请求的情况。  • 为高吞吐量的producer，调优缓冲区的大小，特别是buffer.memory和batch.size(以字节为单位)。由于batch.size是按照分区设定的，而producer的性能和内存的使用量，都可以与topic中的分区数量相关联。因此，此处的设定值将取决于如下几个因素：producer数据速率(消息的大小和数量)、要生成的分区数、以及可用的内存量。请记住，将缓冲区调大并不总是好事，如果producer由于某种原因而失效了(例如，某个leader的响应速度比确认还要慢)，那么在堆内内存(on-heap)中的缓冲的数据量越多，其需要回收的垃圾也就越多。  • 检测应用程序，以跟踪诸如生成的消息数、平均消息大小、以及已使用的消息数等指标。  针对Brokers的最佳实践  • 在各个brokers上，请压缩topics所需的内存和CPU资源。日志压缩需要各个broker上的堆栈(内存)和CPU周期都能成功地配合实现。而如果让那些失败的日志压缩数据持续增长的话，则会给brokers分区带来风险。您可以在broker上调整log.cleaner.dedupe.buffer.size和log.cleaner.threads这两个参数，但是请记住，这两个值都会影响到各个brokers上的堆栈使用。如果某个broker抛出OutOfMemoryError异常，那么它将会被关闭、并可能造成数据的丢失。而缓冲区的大小和线程的计数，则取决于需要被清除的topic partition数量、以及这些分区中消息的数据速率与密钥的大小。对于Kafka的0.10.2.1版本而言，通过ERROR条目来监控日志清理程序的日志文件，是检测其线程可能出现问题的最可靠方法。  • 通过网络吞吐量来监控brokers。请监控发向(transmit，TX)和收向(receive，RX)的流量，以及磁盘的I/O、磁盘的空间、以及CPU的使用率，而且容量规划是维护群集整体性能的关键步骤。  • 在群集的各个brokers之间分配分区的leader关系。Leader通常会需要大量的网络I/O资源。例如，当我们将复制因子(replication factor)配置为3、并运行起来时，leader必须首先获取分区的数据，然后将两套副本发送给另两个followers，进而再传输到多个需要该数据的consumers上。因此在该例子中，单个leader所使用的网络I/O，至少是follower的四倍。而且，leader还可能需要对磁盘进行读操作，而follower只需进行写操作。  • 不要忽略监控brokers的in-sync replica(ISR)shrinks、under-replicated partitions和unpreferred leaders。这些都是集群中潜在问题的迹象。例如，单个分区频繁出现ISR收缩，则暗示着该分区的数据速率超过了leader的能力，已无法为consumer和其他副本线程提供服务了。  • 按需修改Apache Log4j的各种属性。Kafka的broker日志记录会耗费大量的磁盘空间，但是我们却不能完全关闭它。因为有时在发生事故之后，需要重建事件序列，那么broker日志就会是我们最好的、甚至是唯一的方法。  • 禁用topic的自动创建，或针对那些未被使用的topics建立清除策略。例如，在设定的x天内，如果未出现新的消息，您应该考虑该topic是否已经失效，并将其从群集中予以删除。此举可避免您花时间去管理群集中被额外创建的元数据。  • 对于那些具有持续高吞吐量的brokers，请提供足够的内存，以避免它们从磁盘子系统中进行读操作。我们应尽可能地直接从操作系统的缓存中直接获取分区的数据。然而，这就意味着您必须确保自己的consumers能够跟得上“节奏”，而对于那些延迟的consumer就只能强制broker从磁盘中读取了。  • 对于具有高吞吐量服务级别目标(service level objectives，SLOs)的大型群集，请考虑为brokers的子集隔离出不同的topic。至于如何确定需要隔离的topics，则完全取决于您自己的业务需要。例如，您有一些使用相同群集的联机事务处理(multiple online transaction processing，OLTP)系统，那么将每个系统的topics隔离到不同brokers子集中，则能够有助于限制潜在事件的影响半径。  • 在旧的客户端上使用新的topic消息格式。应当代替客户端，在各个brokers上加载额外的格式转换服务。当然，最好还是要尽量避免这种情况的发生。  • 不要错误地认为在本地主机上测试好broker，就能代表生产环境中的真实性能了。要知道，如果使用复制因子为1，并在环回接口上对分区所做的测试，是与大多数生产环境截然不同的。在环回接口上网络延迟几乎可以被忽略的，而在不涉及到复制的情况下，接收leader确认所需的时间则同样会出现巨大的差异。  **金融项目：**  开户模块：注册的时候需要填写一些个人信息，账户填写完成之后会有一个离焦事件，查看数据库有没有相同的用户名，如果有的话返回提示信息，如果没有则显示对勾，如果密码或者一些加\*的信息没有填写则会显示提示信息，信息填写完成后提交注册信息，后台同时验证，验证通过后数据库创建用户个人信息，包含用户各类状态属性，以及版本号，实名认证属性，个人基本资料，邮箱手机号等等，登陆成功之后会给银行放送当前个人信息，调用银行接口，银行根据当前个人信息创建个人虚拟账户，如果用户想要投标，那么需要充值，用户必须在银行页面上绑定银行卡进行转账到个人虚拟账户，避免平台非法集资。（当用户完成平台注册之后，使用httpclient调用存管银行接口，会让用户在存管银行进行实名认证，需要用户填写要绑定的银行卡，真实姓名等相关信息。存管银行会为用户开通存管账户。之后的用户借款、投资、充值、提现都是通过存管银行进行资金流转）  发标模块：比如说实名认证，风控资料，还需要与后台管理人员进行视频认证，视频认证主要是用来核对用户的实名认证信息是否正确，并且审核人员会对借款人进行一些提问，另外，在视频认证的时候，借款人会签署一份借款承诺书，这个借款承诺书会通过视频的录屏，签字等作为借款一个辅助依据，在信用贷中，列出了申请信用贷的几个条件，必须要满足这几个条件，才能够去填写申请贷款的界面，如果当前是登录用户，就可以马上看到当前用户针对这几个条件的满足与否，如果哪个条件没有满足，后面都会有个×，并且点击相应的连接，就能跳转到做这个认证的界面，如果有任何一个条件没有满足或者当前没有登陆，点击申请贷款都没有任何反应，填写成功以后显示都是对勾那么就可以点击申请贷款了，到达贷款页面以后会有几个需要填写的信息，借款金额即本次申请的贷款额度，这个贷款额度的范围为：最小贷款金额<= 本次申请贷款额度 <= 当前剩余信用额度，借款利息即户填写本次借款想提供给投资人的利息（年化利率），一般系统中对年化利率也会有一个范围控制（3%~20%）借款期限是下拉列表，用户可以选择借款期限，一般选择1,3,6,9,12个月，选择多少个月，相当于借款就按照多少个月来分期还款，还款方式有两种方式，一种是按月分期；按等额本息法还款，每期的还款金额相同，另一种是按月到期；每月返还利息，到期返还本金。最小投标：投资人要投这个标，最小的投标金额，如果借款人不填写，系统都会规定一个默认的最小投标金额，一般是50，一般投标也有最大投标金额，一个人投标不能超过一个标的50%，招标天数，招标1~5天可选，意思就是招标开始到结束的最大等待招标时间，标的标题和内容，就是贷款的标题和贷款说明，一般在贷款说明说需要说清楚贷款的原因，目的，还款能力等。都填写完成之后点击提交，本标就会进入审核阶段,对你的贷款的原因，目的，还款能力进行审核，只有通过审核才可以开始招标，通过之后会显示你的招标页面显示你填写的信息，还会显示你这个标的所需总共金额，还剩多少金额，自己登陆这个页面还会有一个取消招标，如果在时限内没有把标投满，那么就会进入到流标阶段，投满之后会进入一审和二审阶段，在最初的审核到一审和二审任何一处不通过都会进入到流标阶段，而初审，一审和二审这需要有三个后台管理人员分别审核，防止出现疏漏，全部通过后标里的虚拟货币就会到贷款人平台账户上。  充值模块：用户先在银行页面绑定个人银行卡，设置虚拟账户支付密码，然后从银行卡把资金转入到用户虚拟账户上去，进行充值。（用户通过点击我要充值，进入充值页面，页面分为快捷支付，支付宝充值以及跨行转账，当时快捷充值金额0.15%收取手续费，当时考虑到成本问题，将快捷支付放到了最后一个。当用户点击某一个充值方式时，比如说支付宝充值，回跳到支付宝充值页面，当时做的是一个二维码支付，通过手机操作，填写绑定的银行卡以及要充值的金额。当我们通过支付宝或者其他的充值方式充值成功后，支付宝会发送反馈信息。支付宝会与存管银行对接，将用户的存管账户余额增加，同时平台会调用存管银行的接口，当用户的账户余额增加，平台就会将用户对应的虚拟账户余额增加，这样就完成了充值资金的同步）  提现模块：先确定用户是否绑定银行卡，如果绑定直接可以在用户虚拟账户上进行转账，如果没绑定，要先绑定银行卡。（用户点击我要提现时，会跳转到提现列表，列表上有用户的真实姓名，提现金额，实际到账金额，用户的手续费，用户完成提现申请后，后台会根据用户的提现申请进行审核，首先当前用户不能有多个提现申请，还有就是当前实际提现金额不能大于账户的可用余额，当审核通过后，会提示用户提现申请审核成功，跳转到存管银行的提现页面，页面上同样也有用户的真实姓名，提现金额，实际到账金额，用户的手续费以及银行卡号等。提现成功后，存管银行会将用户的存管账户钱直接打到用户提供的银行卡上，平台实时监控用户账户资金变化，通过存管银行反馈消息，修改用户在平台上的虚拟账户资金）  投资模块：选定标，点击投标，先是输入支付密码成功后短信验证，如果资金足够进行业务操作，资金不足会给予提示。  还款模块：到达时限后会给予提示，如果资金足够可以直接进行转账，但是资金不足，就会给予提示如果长期不还将降低或取消额度。（借款成功之后，会生成还款对象和回款对象。借款人根据还款对象来分期，首先判断当前用户账户余额是否大于还款金额，小于则提醒用户账户余额不足，去充值界面充值，假设账户金额大于还款金额，点击还款时需要更新几个表的数据，比如投资人账户可用余额增加，代收本金减少，生成账户成功收款流水，若是最后一期，则还款金额应该等于总金额减已经还款的金额，同时还款人的还款状态也要取消等）  后台管理模块：后台人工审核身份证，风控资料，标的初审，一审和二审，查看用户登陆情况，是否账号异常 。  关键数据加密，MySQL的AES\_ENCRYPT和DES\_ENCRYPT函数对字符串进行MD5加密，数据库防篡改,代码中使用hashcode直接验证,Quartz定时查询数据库验。  问题：**借款信息是怎么展示的？**  回答：我们在项目中首页展示投标信息时用到了redis。因为投标信息比较多，属于热数据，所以我们当时考虑使用redis来存储。key是string类型的bidrequest字符串，value是一个hash，key为投标信息的id，value为投标信息对象。通过redisTemplate.opsForHash().put()方法存储。当用户投标信息审核成功，准备发布时，会将投标信息存储到redis里面。当在前台页面展示时，会首先redis里面读取，如果redis里面没有，则从mysql里面读取，如果有数据，则将数据存到redis里面并返回，如果没有数据，会在redis里面给对应的key设置一个null值。  问题：**你们是怎么防止别人恶意调用短信接口的？**  回答：我们项目上线之后，线上用的短信接口被攻击了,我们的用户就收到了很多条垃圾短信，然后就被用户投诉,当时借鉴了大多数网站的设计，我们在点击获取验证码按钮之前，需要填写一个图片验证码，来防止脚本恶意攻击。  问题：**redis解决缓存失效和缓存击穿问题？**  回答：首先先说下缓存失效，在同一时间有大量的key同时失效，瞬间加大了mysql的压力。解决办法就是存储key时失效时间要错开。缓存击穿：他指的是用户访问的数据从redis里面查不到，从mysql里面查，如果重复访问，会造成mysql的压力加大。所以就要要求我们第一次从mysql里面查的时候，如果没有数据，就在redis把对应的value设置为null值。  问题：**分布式的架构下怎么保证id是唯一的？**  回答：我们在项目里面使用的是雪花算法。使用的是idword工具类创建id，如果要求不高的话，可以使用时间戳来定义id。  问题：**在集群模式下怎么保证定时任务不会重复执行？**  回答：我们项目里面没有用到定时任务，我就说一下我的看法，我们在定时任务执行之后，我们会项目的名称，和ip以key-value形式存到redis里面，并设置合理的过期时间，这样当其他定时任务出发之前，就先要从redis里面判断，是否存在当前的key-value，如果存在，就不让定时任务执行。这样就可以解决重复执行的问题。  问题：**验证码的消息是怎么获取到的？**  回答：我们把用户的手机号，随机生成的四位验证码和在阿里云注册时的签名、模板等参数存到map里面，通过rabbittemplate存到RabbitMQ里面，消费者通过RabbitListener注解监听生产者，一旦监听到消息，获得到参数，就调用阿里大于的短信接口，完成发送短信的功能。  问题：**手机发送验证码？**  回答：当用户点击我要注册时，会跳转到注册页面，页面上需要输入用户的手机号，获取验证码，我们使用了阿里大于和云通讯两个短信平台来实现发送验证码。就比如说阿里大于，我们调用阿里大于接口，传递手机号，随机的四位验证码，和在阿里云注册时的签名、模板等参数，来完成验证码的发送。为了防止用户多次点击获取验证码，我们把验证码以string类型形式存储到redis中，key是用户的手机号，value是获取的验证码。存储redis有四个参数，key、value、过期时间和时间类型（小时、分钟、秒等）我们当时设置的验证码过期时间为60秒。在60秒之内用户不可以对同一个手机号重复发送验证码。当用户输入验证码之后，会向后台发送请求，判断验证码是否正确。因为当前验证码存到了redis里面，所以我们从redis里面获取验证码，同时设置了过期时间，所以也能防止用户输入过期的验证码。  问题：**用户的安全登录时怎么实现的？**  回答：我们使用springsecurity实现用户的安全登录。编写配置文件spring-security.xml里面。我记得有个http的标签，可以对静态资源放行，并设置登陆成功的跳转页面，默认的登录页面。当用户发送登录请求时，通过配置文件。我们自定义一个认证类，实现UserDetailsService接口。重写loadUserByUserName方法，然后创建角色列表，将在配置文件里面配置的角色放到列表里面，根据用户名去数据库查询是否存在该用户，如果存在将用户的密码与输入的密码进行比对，返回userDetails。这样就完成了安全登录。 |
| 人资问题 |
| 问题：**你介绍一下自己？**  回答：面试官您好，我叫史淋伯，今年25岁，来自于山西临汾，就业于java开发已经3年多了，今天来贵公司是应聘java开发工程师职位的。我在上一家公司职位是JAVA开发工程师，主要的工作职责是对项目模块的设计开发，做过金融类的P2P铜钱贷项目和广告类的点阅广告平台项目。最近做的是点阅广告平台项目，我们项目组采用了springcloud的服务治理方案，运用了eureka注册中心、zuul网关、feign声明式事务调用、hystrix断路器等组件。这个广告平台系统包括广告投放系统、广告检索系统、扣费系统、报表系统等模块，我在其中主要负责广告投放系统和广告检索系统这两个模块。以上就是我的简单自我介绍，希望接下来可以与各位有个愉快的聊天。  问题：**你得离职原因是啥？**  回答：我希望有一个好的学习和成长的发展平台。  问题：**你觉得自己的优点是？你觉得自己有啥缺点？**  回答：有良好的编程习惯，具有良好沟通能力、问题解决能力、能够承担较强的工作压力，善于钻研、乐于分享、具有良好的团队合作精神，敢于挑战、不断战胜自我，性格乐观、能保持积极的态度对待工作。  对待工作太较劲了。  问题：**上家公司的同事和领导是怎么评价你的？**  回答：我的同事和领导说我能保持积极的态度对待工作。  问题：**介绍下你的上家公司是做哪块的吧？**  回答：北软互联（北京）科技有限公司于2017年10月19日成立。法定代表人王长智,公司经营范围包括：信息科技、计算机科技、电子科技、计算机软硬件、网络科技、光机电一体化、环保科技技术领域内的技术开发、技术咨询、技术服务、技术转让；销售计算机、软件及辅助设备、电子产品、机械设备、文化用品、服装、体育用品；企业形象策划（不含中介服务）；组织文化艺术交流活动（不含演出）；会议服务；承办展览展示活动；电脑图文设计、制作；设计、制作、代理、发布广告；翻译服务；市场调查等。  问题：**在上家公司你做了哪些项目？**  回答：我上家公司是北软互联科技有限公司，这是一家外包公司。我具体的参加了两个项目的开发，这两个项目是点阅广告系统和铜钱贷。  问题：**你在项目中是什么角色？**  回答：我在项目中具体的负责后台接口的开发。  问题：**你大学在哪上的？**  回答：山西工商学院。  问题：**你的大学专业是什么？**  回答：计算机科学与技术。  问题：**具体学的那些科目？**  回答：C语言程序设计、电子技术、数据结构、操作系统、数据库原理与应用、计算机组成原理、计算机网络、Java开发、大数据处理与分析、三维建模技术、嵌入式应用与开发。  问题：**可以说一下你的上学与工作经历吗?**  回答：2019.05.01( 25 )-工作-2015.07.01 (21)-大学-2011.09.01(17)-高中-2008.09.01(14）-初中-2005.09.01（11）  -小学-2000.09.01（6）学前1994.02.11  问题：**你说一下你上家公司？**  回答:北软互联（北京）科技有限公司于2017年10月19日成立。法定代表人王长智,公司经营范围包括：信息科技、计算机科技、电子科技、计算机软硬件、网络科技、光机电一体化、环保科技技术领域内的技术开发、技术咨询、技术服务、技术转让；销售计算机、软件及辅助设备、电子产品、机械设备、文化用品、服装、体育用品；企业形象策划（不含中介服务）；组织文化艺术交流活动（不含演出）；会议服务；承办展览展示活动；电脑图文设计、制作；设计、制作、代理、发布广告；翻译服务；市场调查等。  问题：**公司在哪？**  回答：北京市昌平区东小口镇立汤路30号71号楼2层20。  问题：**你对于加班有什么看法？**  回答：我感觉有利于锻炼我的能力，使自己的各方面能力有更快的提升，在这个竞争激烈的时代我认为是有必要的。并且对于企业而言，我认为合理的加班也有利于企业的快速发展，使个人和企业之间达到共赢。  问题：**假如公司遇到困境，你该如何做？**  回答：我会和公司同舟共济。  问题：**说一下你3到5年的职业规划？**  回答：假如有幸入职贵公司，我会首先熟悉公司业务和技术，以及自己的岗位职责，达到胜任自己的岗位，并且在工作之余结合公司的发展学习新技术，使自己更加的优秀。  问题：**你上家公司的薪资是多少？**  回答：14K    问题：**你的期望薪资是多少？**  回答：16K |