# 云计算 1607C 晋相瑞

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

有趣广告系统

**// 北京昊月科技有限公司，昌平区沙河沙阳路18号**

**// 北京步峰科技有限公司，海淀区上地东里一区4号楼220室**

1. 功能模块介绍(按照自己得实际项目写)

**广告投放**：广告主可以在该系统进行推广计划、推广单元以及创意的创建与修改。

每个推广计划下包含多个推广单元，推广单元与推广创意是多对多的关系。当广告主进行一些计划操作时，接收前台传来的数据进行处理，同时发送到Kafka中，并返回状态码。（使用Postman测试数据）

**检索系统**：广告数据索引的设计，推广计划、推广单元、推广创意通过正向索引的设计根据唯一键找到对应的对象。推广单元的地域限制、关键词限制以及兴趣限制通过倒排索引的设计，即根据内容确定包含关系的对象。当媒体方检索符合条件的创意时，从前台传来媒体方标识、请求的基本信息、请求的匹配信息，后台进行检索过滤得到符合条件的创意返回url链接。

监听Kafka将增量数据同步到ElasticSearch中，当媒体方检索符合条件的创意时，从前台传来媒体方标识、请求的基本信息、请求的匹配信息，后台从ElasticSearch进行检索过滤得到符合条件的创意返回url链接。

项目介绍

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。

1. 开发中遇到的技术难点以及解决办法

1、全量加载

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。

//难点2：增量加载：

//创建一个新的线程，监听Binlog日志文件，通过event事件来判断增删改，

1. 关于JWT

<**dependency**>  
 <**groupId**>io.jsonwebtoken</**groupId**>  
 <**artifactId**>jjwt</**artifactId**>  
 <**version**>0.9.0</**version**>  
</**dependency**>

3、Spring Cloud简介

Spring Cloud是一个基千SpringBoot实现的微服务架构开发 工具。它为微服务架构中

涉及的 配置管理、服务治理、 断路器、 智能路由、微代理、 控制总线、 全局锁、 决策竞选、分布式会话和集群状态管理等操作提供了一种简单的开发方式。

4、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。

5、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

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+……）

1. 项目技术亮点分析

SpringCloud + SpringBoot + SpringData JPA + ElasticSearch + Kafka

一、ElasticSearch的存储数据过程

* 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。

### es 读数据过程

可以通过 doc id 来查询，会根据 doc id 进行 hash，判断出来当时把 doc id 分配到了哪个 shard 上面去，从那个 shard 去查询。

* 客户端发送请求到**任意**一个 node，成为 coordinate node。
* coordinate node 对 doc id 进行哈希路由，将请求转发到对应的 node，此时会使用 round-robin **随机轮询算法**，在 primary shard 以及其所有 replica 中随机选择一个，让读请求负载均衡。
* 接收请求的 node 返回 document 给 coordinate node。
* coordinate node 返回 document 给客户端。
* 客户端发送请求到一个 coordinate node
* 协调节点将搜索请求转发到**所有**的 shard 对应的 primary shard 或 replica shard，都可以。
* query phase：每个 shard 将自己的搜索结果（其实就是一些 doc id）返回给协调节点，由协调节点进行数据的合并、排序、分页等操作，产出最终结果。
* fetch phase：接着由协调节点根据 doc id 去各个节点上**拉取实际**的 document 数据，最终返回给客户端。

写请求是写入 primary shard，然后同步给所有的 replica shard；读请求可以从 primary shard 或 replica shard 读取，采用的是随机轮询算法。

1. 项目中为什么要使用kafka？

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。

1. 如何保证kafka的高可用？

### Kafka 的高可用性

Kafka 一个最基本的架构认识：由多个 broker 组成，每个 broker 是一个节点；你创建一个 topic，这个 topic 可以划分为多个 partition，每个 partition 可以存在于不同的 broker 上，每个 partition 就放一部分数据。

这就是**天然的分布式消息队列**，就是说一个 topic 的数据，是**分散放在多个机器上的，每个机器就放一部分数据**。

Kafka 0.8 以后，提供了 HA 机制，就是 replica（复制品） 副本机制。每个 partition 的数据都会同步到其它机器上，形成自己的多个 replica 副本。所有 replica 会选举一个 leader 出来，那么生产和消费都跟这个 leader 打交道，然后其他 replica 就是 follower。写的时候，leader 会负责把数据同步到所有 follower 上去，读的时候就直接读 leader 上的数据即可。只能读写 leader？很简单，**要是你可以随意读写每个 follower，那么就要 care 数据一致性的问题**，系统复杂度太高，很容易出问题。Kafka 会均匀地将一个 partition 的所有 replica 分布在不同的机器上，这样才可以提高容错性。

这么搞，就有所谓的**高可用性**了，因为如果某个 broker 宕机了，没事儿，那个 broker上面的 partition 在其他机器上都有副本的，如果这上面有某个 partition 的 leader，那么此时会从 follower 中**重新选举**一个新的 leader 出来，大家继续读写那个新的 leader 即可。这就有所谓的高可用性了。

**写数据**的时候，生产者就写 leader，然后 leader 将数据落地写本地磁盘，接着其他 follower 自己主动从 leader 来 pull 数据。一旦所有 follower 同步好数据了，就会发送 ack 给 leader，leader 收到所有 follower 的 ack 之后，就会返回写成功的消息给生产者。（当然，这只是其中一种模式，还可以适当调整这个行为）

**消费**的时候，只会从 leader 去读，但是只有当一个消息已经被所有 follower 都同步成功返回 ack 的时候，这个消息才会被消费者读到。

1. 保证数据不被重复消费

在kafka中每个消息在写进去的时候都会有一个offset，代表消息的序号，然后消费者消费数据，会定时定期的向kafka提交offset，告诉kafka这条消息我已经消费过了，我下次消费的时候就不会消费这条数据了。

1. 如何解决kafka丢失数据问题？
2. 消费者弄丢了数据，也就是消费者刚拿到数据后自动提交了offset，kafka以为你已经消费好了这条数据，此时还没来的及消费，消费者宕机了，那么这条数据就丢失了。那么我们需要关闭自动提交offset，当消息消费完成后，手动提交offset。

private static KafkaConsumer<String, String> *consumer*;  
private static Properties *properties*;  
  
static {  
  
 *properties* = new Properties();  
  
 *properties*.put("bootstrap.servers", "39.106.113.166:9092");*//连接kafka的地址  
 properties*.put("key.deserializer",  
 "org.apache.kafka.common.serialization.StringDeserializer");  
 *properties*.put("value.deserializer",*//反序列化* "org.apache.kafka.common.serialization.StringDeserializer");  
 *properties*.put("group.id", "test-consumer-group");*//消费者组*}

*properties*.put("auto.commit.offset", false);*//关闭自动提交offset  
consumer* = new KafkaConsumer<>(*properties*);  
*consumer*.subscribe(Collections.*singletonList*("test0"));

*consumer*.commitSync();*//手动提交*

#### 2、Kafka 弄丢了数据

这块比较常见的一个场景，就是 Kafka 某个 broker 宕机，然后重新选举 partition 的 leader。大家想想，要是此时其他的 follower 刚好还有些数据没有同步，结果此时 leader 挂了，然后选举某个 follower 成 leader 之后，不就少了一些数据？这就丢了一些数据啊。

生产环境也遇到过，我们也是，之前 Kafka 的 leader 机器宕机了，将 follower 切换为 leader 之后，就会发现说这个数据就丢了。

所以此时一般是要求起码设置如下 4 个参数：

* 给 topic 设置 replication.factor 参数：这个值必须大于 1，要求每个 partition 必须有至少 2 个副本。
* 在 Kafka 服务端设置 min.insync.replicas 参数：这个值必须大于 1，这个是要求一个 leader 至少感知到有至少一个 follower 还跟自己保持联系，没掉队，这样才能确保 leader 挂了还有一个 follower 吧。
* 在 producer 端设置 acks=all：这个是要求每条数据，必须是**写入所有 replica 之后，才能认为是写成功了**。
* 在 producer 端设置 retries=MAX（很大很大很大的一个值，无限次重试的意思）：这个是**要求一旦写入失败，就无限重试**，卡在这里了。

1. 项目答辩中亮点简历审核评定标准

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **考核内容** | **简历样式是否规范** | **项目内容是否完整** | **项目中是否有亮点** | **技术难点解决办法是否高效** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |
| 简历是否合格 | | | | 合格 | 不合格 |

1. 项目答辩成绩评定标准

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **考核内容** | **亮点简历与项目描述是否匹配** | **项目流程是否清晰** | **学生讲解是否流畅** | **项目介绍是否到位** | **技术亮点考核是否合格** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |  |
| 项目答辩是否合格 | | | | 合格 | 不合格 |  |