# 云计算 1607C 王东旭

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

|  |
| --- |
| 广告管理系统是基于C2C模式的平台，可以理解为目前流行的广告联盟。参与者主要包括广告提供商、站长、app所有者、平台管理员等。  广告提供商主要提供广告、此广告不同于商品，它放在平台上，站长、app所有者会获取广告信息并在自己的网站或者软件上进行推广，推广效果是公开的。平台管理者拥有平台所有权，能对平台进行最高权限管理。 |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| 1. 广告投放系统：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 2. 广告检索系统：媒体方对广告系统发起请求，广告系统能够检索符合要求的广告数据, 设计索引的目的就是为了加快检索的速度，将原始数据抽象，规划出合理的字段，在内存中构建广告数据索引。 3. 曝光监测系统：监测广告数据,进行一个曝光,对下次的广告投放起到了一个很好的借鉴和参考. 4. 扣费系统：广告的每一次曝光都是需要扣费的,且这个系统里面负责了将广告数据置位的功能. 5. 报表系统：构建广告数据报表，比如广告 A 在地域 B 中一共曝光了多少次，主要是OLAP（联机分析处理）的过程 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| **难点1：对分布式事务的解决**  解决方法：分布式事务就是指事务的参与者、支持事务的服务器、资源服务器以及事务管理器分别位于不同的分布式系统的不同节点之上。简单的说，就是一次大的操作由不同的小操作组成，这些小的操作分布在不同的服务器上，且属于不同的应用，分布式事务需要保证这些小操作要么全部成功，要么全部失败。本质上来说，分布式事务就是为了保证不同数据库的数据一致性。  分布式事务的解决方案有如下几种：全局消息、基于可靠消息服务的分布式事务、TCC、最大努力通知。  **难点2：在使用微服务架构时，需要解决点对点方式的缺点**  解决方法：选用一个轻量级的总线，能够提供非业务功能的抽象。这就是API网关方式。在微服务架构中，后端服务往往不直接开放给调用端，而是通过一个服务网关的根据请求的url ，路由到相应的服务，即实现请求转发。业务接口通过API网关暴露，是所有客户端接口的唯一入口，微服务之间的通信也通过API网关，这时我们就使用Zuul,来实现服务网关的功能。  **难点3：在开发过程中，在访问数据库中，可能会出现脏读的现象，如何解决？**  解决方法：脏读又称无效数据的读出，是指在数据库访问中，事务T1将某一值修改，然后事务T2读取该值，此后T1因为某种原因撤销对该值的修改，这就导致了T2所读取到的数据是无效的。为了解决这种情况，SQL标准定义的隔离级别设置为Read Committed（读提交），它满足了隔离的简单定义：一个事务只能看见事务所作的改变。  **难点4：SQL优化**  解决方法：  1.对查询进行优化，要尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。  2.应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描, 最好不要给数据库留NULL，尽可能的使用 NOT NULL填充数据库。  3.应尽量避免在 where 子句中使用 != 或 <> 操作符，否则引擎将放弃使用索引而进行全表扫描。  4.应尽量避免在 where 子句中使用 or 来连接条件，如果一个字段有索引，一个字段没有索引，将导致引擎放弃使用索引而进行全表扫描。可以使用union 替换or。  5.索引并不是越多越好，索引固然可以提高相应的select的效率，但同时也降低了insert及update的效率，因为insert或update时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过6个，若太多则应考虑一些不常使用到的列上建的索引是否有必要  6.尽可能的使用varchar/nvarchar代替char/nchar，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。  7.任何地方都不要使用select \* from t ，用具体的字段列表代替“\*”，不要返回用不到的任何字段。  ……  **难点5：在一个分布式系统里，许多依赖不可避免的会调用失败，比如超时、异常等，如何能够保证在一个依赖出问题的情况下，不会导致整体服务失败**  解决方法：这个就是Hystrix需要做的事情。Hystrix提供了熔断、隔离、回退、缓存、监控等功能，能够在一个、或多个依赖同时出现问题时保证系统依然可用。 |

1. 项目技术亮点分析

|  |
| --- |
| **(1)项目框架：**项目采用得**SpringBoot+Spring Cloud**微服务框架，从Spring的优点上来说，第一是一个轻量级框架，从大小与开销两方面而言spring都是轻量的，此外，Spring是非侵入的；典型的，Spring应用中的对象不依赖于Spring的特定类。第二就是通过IOC促进了低耦合，第三Spring支持面向切面的编程，并且把应用业务逻辑和系统服务分开；Spring Cloud是一个集成了众多开源的框架，利用Spring Boot的开发便利性实现了服务治理、服务注册与发现、负载均衡、数据监控，REST API发布方式等，基本囊括了分布式框架所需要的所有功能。是一套易开放、易部署、易维护的分布式开发工具包。  **(2)服务中心：**项目服务中心用的是Eureka，它提供了完整的Service Registry和Service Discovery实现，是Spring Cloud体系中最重要最核心的组件之一。它将所有的可以提供的服务都注册到它这里来管理，其它各调用者需要的时候去注册中心获取，然后再进行调用，避免了服务之间的直接调用，方便后续的水平扩展、故障转移等，Eureka内部已经提供均衡负载的功能，因此使用了Eureka就自动具有了注册中心、负载均衡、故障转移的功能。  **(3)服务网关**：项目中用到的服务网关是Zuul。在微服务架构模式下，后端服务的实例数一般是动态的，对于客户端而言很难发现动态改变的服务实例的访问地址信息。因此在基于微服务的项目中为了简化前端的调用逻辑，通常会引入API Gateway作为轻量级网关，同时API Gateway中也会实现相关的认证逻辑从而简化内部服务之间相互调用的复杂度。Zuul路由是微服务架构中不可或缺的一部分，提供动态路由，监控，弹性，安全等的边缘服务。  Zuul是Netflix出品的一个基于JVM路由和服务端的负载均衡器。  它的具体作用就是服务转发，接收并转发所有内外部的客户端调用。使用Zuul可以作为资源的统一访问入口，同时也可以在网关做一些权限校验等类似的功能。  **(4)解决雪崩效应**：Hystrix。在微服务架构中通常会有多个服务层调用，基础服务的故障可能会导致级联故障，进而造成整个系统不可用的情况，这种现象被称为服务雪崩效应。 在这种情况下就需要整个服务机构具有故障隔离的功能，避免某一个服务挂掉影响全局，Hystrix就扮演这个角色。Hystrix会在某个服务连续调用N次不响应的情况下，立即通知调用端调用失败，避免调用端持续等待而影响了整体服务。Hystrix间隔时间会再次检查此服务，如果服务恢复将继续提供服务。  **(5)消息中间件：**Kafka。Kafka是一个分布式消息队列。具有高性能、持久化、多副本备份、横向扩展能力。生产者往队列里写消息，消费者从队列里取消息进行业务逻辑，即使是非常普通的硬件Kafka也可以支持每秒数百万的消息。一般在架构设计中起到解耦、削峰、异步处理的作用。  **(6)搜索引擎：**ElasticSearch。ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。 |

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 |  |