# 云计算 1607C 刘忠钰

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

|  |
| --- |
| 球约么是一款基于Spring Cloud微服务架构开发的一款篮球爱好者交流平台，采用前后端分离开发，后端采用SpringBoot+SpringData+StringMVC+JPA框架进行开发。它采用关系型数据库MySQL对重要数据进行保存，采用非关系型数据库Redis+MongDB，主要对文章、评论和点赞进行缓存，减轻MySQL压力，记录用户当前状态，防止重复点赞，利用Elasticsearch进行搜索，寻找喜欢的文章，战队和赛事等。它主要面对篮球爱好者。通过平台约战系统，篮球爱好者可以随时发现身边的比赛，并可以进行有酬参与裁判，在平台中使用交友系统可找到“球”味相同的朋友一起讨论篮球中的趣事，在平台上可以使用聊天动态系统分享自己的篮球知识，和精彩瞬间等，使用平台的建立战队系统，可以建立属于自己的战队，和朋友和队友一起玩球，还可以使用招募系统，去招募强力队员，加强战队实力，通过用户微服务进入个人中心，查看个人信息，也可以进行修改。球约么是以社交为切入点，打造一个篮球移动社区平台。 |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| 招募微服务：主要分为球队信息和招募信息，根据热门状态获取球队信息，推送热门球队，进入数据库中进行查询根据球队的信息比如球队实力、战绩、关注数等信息判断是否为热门状态，进行推送，还会根据查询招募信息的状态和创建日期降序排序，分别推送热门招募信息和最近招募信息。热门招募信息会根据它的关注数和发布招募信息的球队的信息判断招募信息的状态是否为热门，一般推送4个热门招募信息，最新招募信息是根据招募信息创建日期降序排序，一般查询前12个招募信息推送为最新招募信息，这些信息数据一般存储在MySQL数据库中。  聊天动态微服务：主要分为聊天和动态两个模块，聊天模块实现的功能是根据两个用户的关系分为临时的陌生人聊天和好友聊天，还有就是战队队员之间的多人聊天，好友之间的多人聊天，这里我主要负责动态的发布，你可以使用动态功能，发布一些精彩瞬间、关于篮球文章、篮球技巧分享等，发布文章之后，我们后台会进行审核改动态是否符合要求，审核通过，即可发布成功，审核失败，就驳回，这里我利用了Redis技术对发表的动态进行缓存，因为使用到了Spring全家桶对项目进行开发,所以使用的是SpringDataRedis，它是对Redis底层进行了高度封装，使用RedisTemplate提供了Redis各种操作，提高查询效率，减轻数据库压力，使用Redis进行缓存是因为，动态会经常被访问，并且数据量比较大，如果总是直接去MySQL数据库中查询，不仅查询速度慢而且会对数据库造成较大压力，使用Redis进行缓存后既能够快速查询，给用户一个很好地体验，也减轻了数据库的压力，其中我通过SpringCache实现数据缓存到Redis中的，因为Spring Cache的核心就是对某个方法进行缓存，其实质就是缓存该方法的返回结果，并把方法参数和结果用键值对的方式存放到缓存中，当再次调用该方法使用相应的参数时，就会直接从缓存里面取出指定的结果进行返回。（常用注解@Cacheable-------使用这个注解的方法在执行后会缓存其返回结果。@CacheEvict--------使用这个注解的方法在其执行前或执行后移除Spring Cache中的某些 元素。）。这里还有一个点赞的功能，让用户可以对喜欢的动态进行点赞，这里我通过Redis实现了一个用户不能重复点赞的功能，当用户点赞时会进入Redis中根据用户ID查看是否已经点过赞，如果没有，点赞加一，如果有，就给用户返回一个不能重复点赞的提示。  评论和吐槽微服务：它是分为评论和吐槽两个模块，评论模块是实现让用户对发布的动态进行评论，分享自己的观点，吐槽模块主要是实现让用户可以对动态、问题、赛事进行吐槽，抒发自己的看点，这里对评论和吐槽是使用MongoDB来进行存储的，是因为吐槽和评论这两个功能数据量大，写入操作频繁，价值较低，而MongoDB更适合来存储这类数据。  搜索微服务：这个模块主要是利用Elasticsearch来实现搜索功能，用户使用搜索功能可以搜索自己想要查看的信息，比如招募信息、球队信息、动态、赛事等  项目中使用的是Spring Data Elasticsearch，他是对Elasticsearch的底层进行了封装，可以实现Elasticsearch同样的功能，ElasticSearch是一个基于Lucene的搜索服务器，这里我使用Head插件实现对Elasticsearch管理，Head插件是通过图形化界面来实现Elasticsearch的日常管理，使用Head插件可以实现对文档内容的删改和查询，还可以创建索引加快查询速度，在Elasticsearch中还使用到了IK中文分词器实现中文分词，IK提供了两个分词算法ik\_smart 和 ik\_max\_word，其中 ik\_smart 为最少切分，ik\_max\_word为最细粒度划分，实现了更全面的搜索，然后对于Elasticsearch与MySQL的数据同步是使用Logstash实现的。  基础微服务：实现基本的增删改查，不用重复在每个微服务写这些代码，统一写完直接调用，简洁代码，提高效率，实现公共异常处理，是为了使我们的代码更容易维护。  用户微服务：主要分为用户登陆、用户注册和用户信息展示，用户登陆时，如果用户没有注册就会提示用户注册，然后跳转到注册页面完成注册，在填写密码时会对密码进行加密，利用BCrypt实现对密码的加密，注册时会发送短信验证码，完成验证，登陆成功后可以进入到个人信息界面，查看和修改个人信息。  短信微服务：主要是实现短信的发送，使用的是阿里大于来实现的短信发送，这里使用到了RabbitMQ，从RabbitMQ中提取消息，调用阿里大于短信接口实现短信发送，这里实际做的就是消息的消费者，RabbitMQ进行对发送短信的监听。  注册中心：利用SpringCloud的五大神兽之一的Eureka实现服务与服务之间的调用，其他微服务注册到Eureka注册微服务，主要是利用Feign实现服务间的调用，Feign可以实现负载均衡减轻服务的压力。 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| 难点1：在项目中实现Redis缓存时，如果Redis挂掉了，或者对缓存数据设置相同的过期时间，导致某段时间内缓存失效，请求全部走数据库。（缓存雪崩）  解决方法：首先这是Redis出现了缓存雪崩问题。对于“对缓存数据设置相同的过期时间，导致某段时间内缓存失效，请求全部走数据库”，这个情况，解决的方法是在缓存的时候给过期时间加上一个随机值，这样就会大幅度减少缓存在同一时间过期，对于“Redis挂掉了，请求全部走数据库”这种情况，解决方法是，如果在事发前，实现Redis的高可用（主从架构+Sentinel 或者Redis Cluster），尽量避免Redis挂掉这种情况发生，如果在事发中，万一Redis真的挂掉了，我们可以设置本地缓存+限流尽量避免我们的数据库被干掉（起码能保证我们的服务还是能正常工作的），如果在事发后，Redis持久化，重启后自动从磁盘上加载数据，快速恢复缓存数据。  难点2：在项目出现请求的数据在缓存大量不命中，导致请求走数据库（缓存穿透）  解决方法1：由于请求的参数是不合法的（每次都请求不存在的参数），于是我们可以使用布隆过滤器（BloomFilter）或者压缩filter提前拦截，不合法就不让这个请求到数据库层。  解决方法2：当我们从数据库找不到的时候，我们也将这个空对象设置到缓存里边去。下次再次请求的时候，就可以从缓存里边获取了。这种情况我们一般会将空对象设置一个较短的过期时间。  难点3：Elasticsearch中的节点，出现平均的选了两个master的情况（比如节点共20个，其中的10个选了一个master，另外10个选了另一个master）  解决方法1：当集群master候选数量不小于3个时，可以通过设置最少投票通过数量超过所有候选节点一半以上来解决脑裂问题  解决方法2：当候选数量为两个时，只能修改为唯一的一个master候选，其他作为data节点，避免脑裂问题。  难点4：客户端在和集群连接时，如何选择特定的节点执行请求  解决方法：TransportClient利用transport模块远程连接一个elasticsearch集群。它并不加入到集群中，只是简单的获得一个或者多个初始化的transport地址，并以 **轮询** 的方式与这些地址进行通信。  难点5：如何监控Elasticsearch集群状态  解决方法：Marvel 让你可以很简单的通过 Kibana 监控 Elasticsearch。你可以实时查看你的集群健康状态和性能，也可以分析过去的集群、索引和节点指标。 |

1. 项目技术亮点分析

|  |
| --- |
| 1. 项目架构：项目采用得微服务架构模式，使用微服务架构模式，个体服务能被   更快地开发，并更容易理解与维护。这种架构使得每个服务都可以由一个团队独立专注开发。开发者可以自由选择任何符合服务 API 契约的技术。微服务架构模式可以实现每个微服务独立部署。开发人员根本不需要去协调部署本地变更到服务。这些变更一经测试即可立即部署。微服务架构模式使得持续部署成为可能。微服务架构模式使得每个服务能够独立扩展。您可以仅部署满足每个服务的容量和可用性约束的实例数目。此外，您可以使用与服务资源要求最匹配的硬件。  (2)项目中我使用了RabiitMQ来实现:解耦、异步、削峰  解耦:如果使用 MQ，A 系统产生一条数据，发送到 MQ 里面去，哪个系统需要数据自己去 MQ 里面消费。如果新系统需要数据，直接从 MQ 里消费即可；如果某个系统不需要这条数据了，就取消对 MQ 消息的消费即可。这样下来，A 系统压根儿不需要去考虑要给谁发送数据，不需要维护这个代码，也不需要考虑人家是否调用成功、失败超时等情况。  异步:  一般互联网类的企业，对于用户直接的操作，一般要求是每个请求都必须在 200 ms 以内完成，对用户几乎是无感知的。  如果**使用 MQ**，那么 A 系统连续发送 3 条消息到 MQ 队列中，假如耗时 5ms，A 系统从接受一个请求到返回响应给用户，总时长是 3 + 5 = 8ms，对于用户而言，其实感觉上就是点个按钮，8ms 以后就直接返回了，爽！网站做得真好，真快！  削锋:  每天 0:00 到 12:00，A 系统风平浪静，每秒并发请求数量就 50 个。结果每次一到 12:00 ~ 13:00 ，每秒并发请求数量突然会暴增到 5k+ 条。但是系统是直接基于 MySQL 的，大量的请求涌入 MySQL，每秒钟对 MySQL 执行约 5k 条 SQL。  一般的 MySQL，扛到每秒 2k 个请求就差不多了，如果每秒请求到 5k 的话，可能就直接把 MySQL 给打死了，导致系统崩溃，用户也就没法再使用系统了。  但是高峰期一过，到了下午的时候，就成了低峰期，可能也就 1w 的用户同时在网站上操作，每秒中的请求数量可能也就 50 个请求，对整个系统几乎没有任何的压力。  如果使用 MQ，每秒 5k 个请求写入 MQ，A 系统每秒钟最多处理 2k 个请求，因  为 MySQL 每秒钟最多处理 2k 个。A 系统从 MQ 中慢慢拉取请求，每秒钟就拉取  2k 个请求，不要超过自己每秒能处理的最大请求数量就 ok，这样下来，哪怕是  高峰期的时候，A 系统也绝对不会挂掉。而 MQ 每秒钟 5k 个请求进来，就 2k 个  请求出去，结果就导致在中午高峰期（1 个小时），可能有几十万甚至几百万的  请求积压在 MQ 中。  这个短暂的高峰期积压是 ok 的，因为高峰期过了之后，每秒钟就 50 个请求进 MQ，但是 A 系统依然会按照每秒 2k 个请求的速度在处理。所以说，只要高峰期一过，A 系统就会快速将积压的消息给解决掉。  (3)断点传续方案  通常视频文件都比较大，所以对于媒资系统上传文件的需求要满足大文件的上传要求。http协议本身对上传文件大  小没有限制，但是客户的网络环境质量、电脑硬件环境等参差不齐，如果一个大文件快上传完了网断了，电断了没  有上传完成，需要客户重新上传，这是致命的，所以对于大文件上传的要求最基本的是断点续传。  什么是断点续传：  引用百度百科：断点续传指的是在下载或上传时，将下载或上传任务（一个文件或一个压缩包）人为的划分为几个  部分，每一个部分采用一个线程进行上传或下载，如果碰到网络故障，可以从已经上传或下载的部分开始继续上传  下载未完成的部分，而没有必要从头开始上传下载，断点续传可以提高节省操作时间，提高用户体验性。  (4)两阶段提交协议(2PC)  为解决分布式系统的数据一致性问题出现了两阶段提交协议（2 Phase Commitment Protocol），两阶段提交由  协调者和参与者组成，共经过两个阶段和三个操作，部分关系数据库如Oracle、MySQL支持两阶段提交协议，本节  讲解关系数据库两阶段提交协议  1）第一阶段：准备阶段（prepare）  协调者通知参与者准备提交订单，参与者开始投票。  协调者完成准备工作向协调者回应Yes。  2）第二阶段：提交(commit)/回滚(rollback)阶段  协调者根据参与者的投票结果发起最终的提交指令。  如果有参与者没有准备好则发起回滚指令。 |

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 |  |