初/中/高级 java面试题

面试题包括以下十九个模块：Java 基础、容器、多线程、反射、对象拷贝、Java Web 模块、异常、网络、设计模式、Spring/Spring MVC、Spring Boot/Spring Cloud、Hibernate、Mybatis、RabbitMQ、Kafka、Zookeeper、MySql、Redis、JVM 。

**一、Java 基础**

**1、JDK 和 JRE 有什么区别？**

jre是java程序运行时环境，jdk是java开发包，自带jre

**2、== 和 equals 的区别是什么？**

基本类型：比较的是值是否相同；引用类型：比较的是引用是否相同；

**3、两个对象的 hashCode()相同，则 equals()也一定为 true，对吗？**

不对，两个对象的 hashCode()相同，equals()不一定 true

**4、final 在 java 中有什么作用？**

**5、java 中的 Math.round(-1.5) 等于多少？**

等于 -1

**6、String 属于基础的数据类型吗？**

**7、java 中操作字符串都有哪些类？它们之间有什么区别？**

**8、String str="i"与 String str=new String(“i”)一样吗？**

不一样，因为内存的分配方式不一样。String str="i"的方式，java 虚拟机会将其分配到常量池中；而 String str=new String(“i”) 则会被分到堆内存中。

**9、如何将字符串反转？**

**10、String 类的常用方法都有那些？**

**11、抽象类必须要有抽象方法吗？**

**12、普通类和抽象类有哪些区别？**

**13、抽象类能使用 final 修饰吗？**

**14、接口和抽象类有什么区别？**

**15、java 中 IO 流分为几种？**

**16、BIO、NIO、AIO 有什么区别？**

**17、Files的常用方法都有哪些？**

**二、容器**

**18、java 容器都有哪些？**

**19、Collection 和 Collections 有什么区别？**

**20、List、Set、Map 之间的区别是什么？**

**21、HashMap 和 Hashtable 有什么区别？**

**22、如何决定使用 HashMap 还是 TreeMap？**

**23、说一下 HashMap 的实现原理？**

**24、说一下 HashSet 的实现原理？**

**25、ArrayList 和 LinkedList 的区别是什么？**

**26、如何实现数组和 List 之间的转换？**

**27、ArrayList 和 Vector 的区别是什么？**

**28、Array 和 ArrayList 有何区别？**

**29、在 Queue 中 poll()和 remove()有什么区别？**

**30、哪些集合类是线程安全的？**

**31、迭代器 Iterator 是什么？**

**32、Iterator 怎么使用？有什么特点？**

**33、Iterator 和 ListIterator 有什么区别？**

**34、怎么确保一个集合不能被修改？**

**三、多线程**

**35、并行和并发有什么区别？**

**36、线程和进程的区别？**

**37、守护线程是什么？**

**38、创建线程有哪几种方式？**

**39、说一下 runnable 和 callable 有什么区别？**

**40、线程有哪些状态？**

**41、sleep() 和 wait() 有什么区别？**

**42、notify()和 notifyAll()有什么区别？**

**43、线程的 run()和 start()有什么区别？**

**44、创建线程池有哪几种方式？**

**45、线程池都有哪些状态？**

**46、线程池中 submit()和 execute()方法有什么区别？**

**47、在 java 程序中怎么保证多线程的运行安全？**

**48、多线程锁的升级原理是什么？**

**49、什么是死锁？**

**50、怎么防止死锁？**

**51、ThreadLocal 是什么？有哪些使用场景？**

**52、说一下 synchronized 底层实现原理？**

**53、synchronized 和 volatile 的区别是什么？**

**54、synchronized 和 Lock 有什么区别？**

**55、synchronized 和 ReentrantLock 区别是什么？**

**56、说一下 atomic 的原理？**

**四、反射**

**57、什么是反射？**

**58、什么是 java 序列化？什么情况下需要序列化？**

**59、动态代理是什么？有哪些应用？**

**60、怎么实现动态代理？**

**五、对象拷贝**

**61、为什么要使用克隆？**

**62、如何实现对象克隆？**

**63、深拷贝和浅拷贝区别是什么？**

**六、Java Web**

**64、jsp 和 servlet 有什么区别？**

**65、jsp 有哪些内置对象？作用分别是什么？**

**66、说一下 jsp 的 4 种作用域？**

**67、session 和 cookie 有什么区别？**

**68、说一下 session 的工作原理？**

**69、如果客户端禁止 cookie 能实现 session 还能用吗？**

**70、spring mvc 和 struts 的区别是什么？**

**71、如何避免 sql 注入？**

**72、什么是 XSS 攻击，如何避免？**

**73、什么是 CSRF 攻击，如何避免？**

**七、异常**

**74、throw 和 throws 的区别？**

**75、final、finally、finalize 有什么区别？**

**76、try-catch-finally 中哪个部分可以省略？**

**77、try-catch-finally 中，如果 catch 中 return 了，finally 还会执行吗？**

**78、常见的异常类有哪些？**

**八、网络**

**79、http 响应码 301 和 302 代表的是什么？有什么区别？**

**80、forward 和 redirect 的区别？**

**81、简述 tcp 和 udp的区别？**

**82、tcp 为什么要三次握手，两次不行吗？为什么？**

**83、说一下 tcp 粘包是怎么产生的？**

**84、OSI 的七层模型都有哪些？**

**85、get 和 post 请求有哪些区别？**

**86、如何实现跨域？**

**87、说一下 JSONP 实现原理？**

**九、设计模式**

**88、说一下你熟悉的设计模式？**

**89、简单工厂和抽象工厂有什么区别？**

**十、Spring/Spring MVC**

**90、什么是spring？为什么要使用 spring？**

1、Spring是一个开源的轻量级的Java开发框架，简化应用程序的开发；

2、核心是IOC（控制反转）容器和AOP（切面编程），松耦合，高扩展性；

3、声明式事务（注解事务），比较编程式事务提高了开发效率；

4、粘合剂，除了自身提供的功能，还可以提供其他框架整合能力；

**91、解释一下什么是 aop，及其原理（动态代理）？**

切面编程 概念：

切面（Aspect）：切入系统的一个切面。比如事务、权限、日志等；

连接点（Join point）：也就是可以进行横向切入的位置；

切面通知（Advice）：切面在某个连接点执行的操作（方法前、方法后、方法异常、方法围绕）；

切点（Pointcut）：符合切点表达式的连接点，也就是真正被切入的地方；

实现原理：

AOP是将切面代码以动态织入实现的。底层采用技术为JDK提供的AspectJ动态代理和cglib动态字节码增强技术，最终都是基于代理模式生成一个代理对象。

JDK动态代理：

主要使用到 InvocationHandler 接口和 Proxy.newProxyInstance() 方法，必须要有实现方法；

**92、解释一下什么是 ioc？**

控制反转（IOC），依赖注入（DI）：所谓控制反转就是把对象（bean）和维护对象（bean）之间的关系的权利转移到Sqring容器中去了（ApplicationContext.xml）而程序本身不在维护了；

**93、spring 有哪些主要模块（七大模块）？**

1、Spring Core：Core模块是Spring的核心类库，Spring的所有功能都依赖于该类库，Core主要实现IOC功能，Sprign的所有功能都是借助IOC实现的；

2、AOP：AOP模块是Spring的AOP库，提供了AOP（拦截器）机制，并提供常用的拦截器，供用户自定义和配置；

3、ORM：供对常用的ORM框架的管理和辅助支持，如ibatis持久层框架，其本身并不对ORM进行实现，仅对常见的ORM框架进行封装，并对其进行管理；

4、DAO模块：提供对JDBC的支持，对JDBC进行封装，允许JDBC使用Spring资源，并能统一管理JDBC事物，并不对JDBC进行实现；

5、WEB模块：提供对常见框架如Struts1，WEBWORK（Struts 2），JSF的支持；

6、Context模块：提供框架式的Bean访问方式；

7、MVC模块：提供了一套轻量级的MVC实现；

**94、spring 常用的注入方式有哪些？**

基于构造方法注入；基于setter方法注入；基于@Bean注解注入；

**95、spring 中的 bean 是线程安全的吗？**

Spring容器中的bean本身不具备线程安全特性，要结合具体的作用域来确定，所以要使用全局变量一定要加final关键字，多线程环境可以使用ThreadLocal本地线程池来解决竞争问题，但比较耗内存。

**96、spring 支持几种 bean 的作用域？**

1、singleton:单例，默认作用域。

2、prototype:原型，每次创建一个新对象。

3、request:请求，每次Http请求创建一个新对象，适用于WebApplicationContext环境下。

4、session:会话，同一个会话共享一个实例，不同会话使用不用的实例。

5、global-session:全局会话，所有会话共享一个实例。

**97、spring 自动装配 bean 有哪些方式？**

隐式的bean发现机制和自动装配；在java代码或者XML中进行显示配置；

**98、spring 事务实现方式有哪些？**

1、编程式事务：代码中调用beginTransaction()、commit()、rollback()等方法；

2、声明式事务：注解事务（@Transactional）、事务代理工厂（TransactionProxyFactoryBean）

3、配置事务：动态代理事务（Aspectj AOP）

**99、说一下 spring mvc 运行流程？**

用户请求 **→ 【前端控制器(DispatcherServlet)】→** 经过【**处理器适配器(HandlerMapping)**】，查找具体的Handler结果原路返回给控制器 **→** 控制器调用【**处理器适配器(HandlerAdapter)**】适配并执行Handler **→** 经过【**处理器(Handler)**】处理具体业务后原路返回ModelAndView给控制器 **→** 控制器请求【**视图解析器(ViewResolver)**】将返回的视图解析成真正的视图，进行国际化处理，并原路返回视图给控制器 **→** 控制器调用**【视图渲染(View)】**将视图转换成response响应**→** 响应结果

**100、spring mvc 有哪些组件？**

* DispatcherServlet：中央控制器，把请求给转发到具体的控制类；
* HandlerMapping：映射处理器，负责映射中央处理器转发给controller时的映射策略；
* ModelAndView：服务层返回的数据和视图层的封装类；
* ViewResolver：视图解析器，解析具体的视图；
* Interceptors ：拦截器，负责拦截我们定义的请求然后做处理工作；

**101、spring事务特性？说一下 spring 的事务隔离？spring事务有哪几种传播行为？**

特性：原子性、一致性、隔离性、持久性

隔离级别：

脏读：一个事务读到另一个事务未提交的更新数据。

幻读：一个事务读到了另一个事务已经提交的 insert 的数据导致多次查询结果不一致。例如事务T1对一张表中的全部数据进行了修改，执行了10分钟。在事务T1开始后的1分钟事务T2向表中插入“一条新数据”，执行1秒完成。那么，事务T1执行结束后发现表中数据还有一条没修改的，就好象发生了幻觉一样。

不可重复读：一个事务读到了另一个事务已经提交的update的数据导致多次查询结果不一致。例如事务T1读取某一数据，事务T2读取并修改了该数据，T1为了对读取值进行检验而再次读取该数据，便得到了不同的结果。解决办法：把数据库的事务隔离级别调整到REPEATABLE\_READ（可重复读）。

传播行为：

PROPAGATION\_REQUIRED 如果存在事务，则支持当前事务。如果没有事务则开启一个新的事务

PROPAGATION\_SUPPORTS 如果存在事务，支持当前事务。如果没有事务，则非事务的执行。

**102、**

**103、@Autowired 的作用是什么？** 按照类匹配自动装配bean的一种注解，@Qualifier按照名称。

**十一、Spring Boot/Spring Cloud**

**104、什么是 spring boot？**

在Spring框架这个大家族中，产生了很多衍生框架，比如 Spring、SpringMvc框架等，Spring的核心内容在于控制反转(IOC)和依赖注入(DI),所谓控制反转并非是一种技术，而是一种思想，在操作方面是指在spring配置文件中创建，依赖注入即为由spring容器为应用程序的某个对象提供资源，比如 引用对象、常量数据等。

SpringBoot是一个框架，一种全新的编程规范，他的产生简化了框架的使用，所谓简化是指简化了Spring众多框架中所需的大量且繁琐的配置文件，所以 SpringBoot是一个服务于框架的框架，服务范围是简化配置文件。

快速构建分布式系统中一些常见模式的工具，例如：

配置管理、服务注册与发现、断路器、智能路由、服务间调用、负载均衡、微代理、控制总线、一次性令牌、全局锁、领导选举、分布式会话、集群状态、分布式消息

**105、为什么要用 spring boot？**

Spring Boot使编码变简单：配置变简单、部署变简单、监控变简单、Spring的不足

**106、spring boot 核心配置文件是什么？**

Spring Boot提供了两种常用的配置文件：properties文件、yml文件

**107、spring boot 配置文件有哪几种类型？它们有什么区别？**

Spring Boot提供了两种常用的配置文件，分别是properties文件和yml文件。相对于properties文件而言，yml文件更年轻，也有很多的坑。可谓成也萧何败萧何，yml通过空格来确定层级关系，使配置文件结构跟清晰，但也会因为微不足道的空格而破坏了层级关系。

**108、spring boot 有哪些方式可以实现热部署？**

①. 使用spring loaded；②. 使用spring-boot-devtools

**109、jpa 和 hibernate 有什么区别？**

**110、什么是 spring cloud？**

Spring Cloud 就是致力于分布式系统、云服务的框架

**111、spring cloud 断路器的作用是什么？**

在Spring Cloud中使用了Hystrix 来实现断路器的功能，断路器可以防止一个应用程序多次试图执行一个操作，即很可能失败，允许它继续而不等待故障恢复或者浪费 CPU 周期，而它确定该故障是持久的。断路器模式也使应用程序能够检测故障是否已经解决，如果问题似乎已经得到纠正，应用程序可以尝试调用操作。

断路器增加了稳定性和灵活性，以一个系统，提供稳定性，而系统从故障中恢复，并尽量减少此故障的对性能的影响。它可以帮助快速地拒绝对一个操作，即很可能失败，而不是等待操作超时（或者不返回）的请求，以保持系统的响应时间。如果断路器提高每次改变状态的时间的事件，该信息可以被用来监测由断路器保护系统的部件的健康状况，或以提醒管理员当断路器跳闸，以在打开状态。

**112、spring cloud 的核心组件有哪些？**

①. 服务发现——Netflix Eureka

一个RESTful服务，用来定位运行在AWS地区（Region）中的中间层服务。由两个组件组成：Eureka服务器和Eureka客户端。Eureka服务器用作服务注册服务器。Eureka客户端是一个java客户端，用来简化与服务器的交互、作为轮询负载均衡器，并提供服务的故障切换支持。Netflix在其生产环境中使用的是另外的客户端，它提供基于流量、资源利用率以及出错状态的加权负载均衡。

②. 客服端负载均衡——Netflix Ribbon

Ribbon，主要提供客户侧的软件负载均衡算法。Ribbon客户端组件提供一系列完善的配置选项，比如连接超时、重试、重试算法等。Ribbon内置可插拔、可定制的负载均衡组件。

③. 断路器——Netflix Hystrix

断路器可以防止一个应用程序多次试图执行一个操作，即很可能失败，允许它继续而不等待故障恢复或者浪费 CPU 周期，而它确定该故障是持久的。断路器模式也使应用程序能够检测故障是否已经解决。如果问题似乎已经得到纠正，应用程序可以尝试调用操作。

④. 服务网关——Netflix Zuul

类似nginx，反向代理的功能，不过netflix自己增加了一些配合其他组件的特性。

⑤. 分布式配置——Spring Cloud Config

这个还是静态的，得配合Spring Cloud Bus实现动态的配置更新。

**十三、Mybatis**

**125、mybatis 中 #{}和 ${}的区别是什么？**

**126、mybatis 有几种分页方式？**

**127、RowBounds 是一次性查询全部结果吗？为什么？**

**128、mybatis 逻辑分页和物理分页的区别是什么？**

**129、mybatis 是否支持延迟加载？延迟加载的原理是什么？**

**130、说一下 mybatis 的一级缓存和二级缓存？**

**131、mybatis 和 hibernate 的区别有哪些？**

**132、mybatis 有哪些执行器（Executor）？**

**133、mybatis 分页插件的实现原理是什么？**

**134、mybatis 如何编写一个自定义插件？**

**十四、RabbitMQ**

**135、rabbitmq 的使用场景有哪些？**

①. 跨系统的异步通信，所有需要异步交互的地方都可以使用消息队列。就像我们除了打电话（同步）以外，还需要发短信，发电子邮件（异步）的通讯方式。

②. 多个应用之间的耦合，由于消息是平台无关和语言无关的，而且语义上也不再是函数调用，因此更适合作为多个应用之间的松耦合的接口。基于消息队列的耦合，不需要发送方和接收方同时在线。在企业应用集成（EAI）中，文件传输，共享数据库，消息队列，远程过程调用都可以作为集成的方法。

③. 应用内的同步变异步，比如订单处理，就可以由前端应用将订单信息放到队列，后端应用从队列里依次获得消息处理，高峰时的大量订单可以积压在队列里慢慢处理掉。由于同步通常意味着阻塞，而大量线程的阻塞会降低计算机的性能。

④. 消息驱动的架构（EDA），系统分解为消息队列，和消息制造者和消息消费者，一个处理流程可以根据需要拆成多个阶段（Stage），阶段之间用队列连接起来，前一个阶段处理的结果放入队列，后一个阶段从队列中获取消息继续处理。

⑤. 应用需要更灵活的耦合方式，如发布订阅，比如可以指定路由规则。

⑥. 跨局域网，甚至跨城市的通讯（CDN行业），比如北京机房与广州机房的应用程序的通信。

**136、rabbitmq 有哪些重要的角色？**

RabbitMQ 中重要的角色有：生产者、消费者和代理：

生产者：消息的创建者，负责创建和推送数据到消息服务器；

消费者：消息的接收方，用于处理数据和确认消息；

代理：就是 RabbitMQ 本身，用于扮演“快递”的角色，本身不生产消息，只是扮演“快递”的角色。

**137、rabbitmq 有哪些重要的组件？**

ConnectionFactory（连接管理器）：应用程序与Rabbit之间建立连接的管理器，程序代码中使用。

Channel（信道）：消息推送使用的通道。

Exchange（交换器）：用于接受、分配消息。

Queue（队列）：用于存储生产者的消息。

RoutingKey（路由键）：用于把生成者的数据分配到交换器上。

BindingKey（绑定键）：用于把交换器的消息绑定到队列上。

**138、rabbitmq 中 vhost 的作用是什么？**

vhost 可以理解为虚拟 broker ，即 mini-RabbitMQ server。其内部均含有独立的 queue、exchange 和 binding 等，但最最重要的是，其拥有独立的权限系统，可以做到 vhost 范围的用户控制。当然，从 RabbitMQ 的全局角度，vhost 可以作为不同权限隔离的手段（一个典型的例子就是不同的应用可以跑在不同的 vhost 中）。

**139、rabbitmq 的消息是怎么发送的？**

首先客户端必须连接到 RabbitMQ 服务器才能发布和消费消息，客户端和 rabbit server 之间会创建一个 tcp 连接，一旦 tcp 打开并通过了认证（认证就是你发送给 rabbit 服务器的用户名和密码），你的客户端和 RabbitMQ 就创建了一条 amqp 信道（channel），信道是创建在“真实” tcp 上的虚拟连接，amqp 命令都是通过信道发送出去的，每个信道都会有一个唯一的 id，不论是发布消息，订阅队列都是通过这个信道完成的。

**140、rabbitmq 怎么保证消息的稳定性？**

提供了事务的功能。通过将 channel 设置为 confirm（确认）模式。

**141、rabbitmq 怎么避免消息丢失？**

消息持久化、ACK确认机制、设置集群镜像模式、消息补偿机制

**142、要保证消息持久化成功的条件有哪些？**

声明队列必须设置持久化 durable 设置为 true；

消息推送投递模式必须设置持久化，deliveryMode 设置为 2（持久）；

消息已经到达持久化交换器；

消息已经到达持久化队列；（以上四个条件都满足才能保证消息持久化成功）

**143、rabbitmq 持久化有什么缺点？**

持久化的缺地就是降低了服务器的吞吐量，因为使用的是磁盘而非内存存储，从而降低了吞吐量。可尽量使用 ssd 硬盘来缓解吞吐量的问题。

**144、rabbitmq 有几种广播类型？**

三种广播模式：

fanout: 所有bind到此exchange的queue都可以接收消息（纯广播，绑定到RabbitMQ的接受者都能收到消息）；  
direct: 通过routingKey和exchange决定的那个唯一的queue可以接收消息；  
topic:所有符合routingKey(此时可以是一个表达式)的routingKey所bind的queue可以接收消息；

**145、rabbitmq 怎么实现延迟消息队列？**

通过消息过期后进入死信交换器，再由交换器转发到延迟消费队列，实现延迟功能；  
使用 RabbitMQ-delayed-message-exchange 插件实现延迟功能。

**146、rabbitmq 集群有什么用？**

集群主要有以下两个用途：

高可用：某个服务器出现问题，整个 RabbitMQ 还可以继续使用；  
高容量：集群可以承载更多的消息量。

**147、rabbitmq 节点的类型有哪些？**

磁盘节点：消息会存储到磁盘。

内存节点：消息都存储在内存中，重启服务器消息丢失，性能高于磁盘类型。

**148、rabbitmq 集群搭建需要注意哪些问题？**

各节点之间使用“–link”连接，此属性不能忽略。

各节点使用的 erlang cookie 值必须相同，此值相当于“秘钥”的功能，用于各节点的认证。

整个集群中必须包含一个磁盘节点。

**149、rabbitmq 每个节点是其他节点的完整拷贝吗？为什么？**

不是，原因有以下两个：

存储空间的考虑：如果每个节点都拥有所有队列的完全拷贝，这样新增节点不但没有新增存储空间，反而增加了更多的冗余数据；  
性能的考虑：如果每条消息都需要完整拷贝到每一个集群节点，那新增节点并没有提升处理消息的能力，最多是保持和单节点相同的性能甚至是更糟。

**150、rabbitmq 集群中唯一一个磁盘节点崩溃了会发生什么情况？**

如果唯一磁盘的磁盘节点崩溃了，不能进行以下操作：不能创建队列、不能创建交换器、不能创建绑定、不能添加用户、不能更改权限、不能添加和删除集群节点

**151、rabbitmq 对集群节点停止顺序有要求吗？**

RabbitMQ 对集群的停止的顺序是有要求的，应该先关闭内存节点，最后再关闭磁盘节点。如果顺序恰好相反的话，可能会造成消息的丢失。

**十五、Kafka**

**152、kafka 可以脱离 zookeeper 单独使用吗？为什么？**

kafka 不能脱离 zookeeper 单独使用，因为 kafka 使用 zookeeper 管理和协调 kafka 的节点服务器。

**153、kafka 有几种数据保留的策略？**

kafka 有两种数据保存策略：按照过期时间保留和按照存储的消息大小保留。

**154、kafka 同时设置了 7 天和 10G 清除数据，到第五天的时候消息达到了 10G，这个时候 kafka 将如何处理？**

这个时候 kafka 会执行数据清除工作，时间和大小不论那个满足条件，都会清空数据。

**155、什么情况会导致 kafka 运行变慢？**

cpu 性能瓶颈、磁盘读写瓶颈、网络瓶颈

**156、使用 kafka 集群需要注意什么？**

集群的数量不是越多越好，最好不要超过 7 个，因为节点越多，消息复制需要的时间就越长，整个群组的吞吐量就越低。  
集群数量最好是单数，因为超过一半故障集群就不能用了，设置为单数容错率更高。

**十六、Zookeeper**

**157、zookeeper 是什么？**

zookeeper 是一个分布式的，开放源码的分布式应用程序协调服务，是 google chubby 的开源实现，是 hadoop 和 hbase 的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

**158、zookeeper 都有哪些功能？**

集群管理：监控节点存活状态、运行请求等。

**159、zookeeper 有几种部署模式？**

单机部署：一台集群上运行；

集群部署：多台集群运行；  
伪集群部署：一台集群启动多个 zookeeper 实例运行。

**160、zookeeper 怎么保证主从节点的状态同步？**

zookeeper 的核心是原子广播，这个机制保证了各个 server 之间的同步。实现这个机制的协议叫做 zab 协议。 zab 协议有两种模式，分别是恢复模式（选主）和广播模式（同步）。当服务启动或者在领导者崩溃后，zab 就进入了恢复模式，当领导者被选举出来，且大多数 server 完成了和 leader 的状态同步以后，恢复模式就结束了。状态同步保证了 leader 和 server 具有相同的系统状态。

**161、集群中为什么要有主节点？**

在分布式环境中，有些业务逻辑只需要集群中的某一台机器进行执行，其他的机器可以共享这个结果，这样可以大大减少重复计算，提高性能，所以就需要主节点。

**162、集群中有 3 台服务器，其中一个节点宕机，这个时候 zookeeper 还可以使用吗？**

可以继续使用，单数服务器只要没超过一半的服务器宕机就可以继续使用。

**163、说一下 zookeeper 的通知机制？**

客户端端会对某个 znode 建立一个 watcher 事件，当该 znode 发生变化时，这些客户端会收到 zookeeper 的通知，然后客户端可以根据 znode 变化来做出业务上的改变。

**十七、MySql**

**164、数据库的三范式是什么？**

第一范式：强调的是列的原子性，即数据库表的每一列都是不可分割的原子数据项。  
第二范式：要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性。  
第三范式：任何非主属性不依赖于其它非主属性。

**165、一张自增表里面总共有 7 条数据，删除了最后 2 条数据，重启 mysql 数据库，又插入了一条数据，此时 id 是几？**

表类型如果是 MyISAM ，那 id 就是 8。  
表类型如果是 InnoDB，那 id 就是 6。

InnoDB 表只会把自增主键的最大 id 记录在内存中，所以重启之后会导致最大 id 丢失。

**166、如何获取当前数据库版本？**

使用 select version() 获取当前 MySQL 数据库版本。

**167、说一下 ACID 是什么？**

Atomicity（原子性）：一个事务（transaction）中的所有操作，或者全部完成，或者全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被恢复（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。即，事务不可分割、不可约简。  
Consistency（一致性）：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设约束、触发器、级联回滚等。

Isolation（隔离性）：数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（Read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（Serializable）。  
Durability（持久性）：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

**168、char 和 varchar 的区别是什么？**

char(n) ：固定长度类型，比如订阅 char(10)，当你输入"abc"三个字符的时候，它们占的空间还是 10 个字节，其他 7 个是空字节。  
chat 优点：效率高；缺点：占用空间；适用场景：存储密码的 md5 值，固定长度的，使用 char 非常合适。

varchar(n) ：可变长度，存储的值是每个值占用的字节再加上一个用来记录其长度的字节的长度。  
所以，从空间上考虑 varcahr 比较合适；从效率上考虑 char 比较合适，二者使用需要权衡。

**169、float 和 double 的区别是什么？**

float 最多可以存储 8 位的十进制数，并在内存中占 4 字节。  
double 最可可以存储 16 位的十进制数，并在内存中占 8 字节。

**170、mysql 的内连接、左连接、右连接有什么区别？**

内连接关键字：inner join；左连接：left join；右连接：right join。

内连接是把匹配的关联数据显示出来；左连接是左边的表全部显示出来，右边的表显示出符合条件的数据；右连接正好相反。

**171、mysql 索引是怎么实现的？**

索引是满足某种特定查找算法的数据结构，而这些数据结构会以某种方式指向数据，从而实现高效查找数据。

具体来说 MySQL 中的索引，不同的数据引擎实现有所不同，但目前主流的数据库引擎的索引都是 B+ 树实现的，B+ 树的搜索效率，可以到达二分法的性能，找到数据区域之后就找到了完整的数据结构了，所有索引的性能也是更好的。

**172、怎么验证 mysql 的索引是否满足需求？**

使用 explain 查看 SQL 是如何执行查询语句的，从而分析你的索引是否满足需求。

explain 语法：explain select \* from table where type=1。

**173、说一下数据库的事务隔离？**

MySQL 的事务隔离是在 MySQL. ini 配置文件里添加的，在文件的最后添加：transaction-isolation = REPEATABLE-READ

可用的配置值：READ-UNCOMMITTED、READ-COMMITTED、REPEATABLE-READ、SERIALIZABLE。

READ-UNCOMMITTED：未提交读，最低隔离级别、事务未提交前，就可被其他事务读取（会出现幻读、脏读、不可重复读）。

READ-COMMITTED：提交读，一个事务提交后才能被其他事务读取到（会造成幻读、不可重复读）。

REPEATABLE-READ：可重复读，默认级别，保证多次读取同一个数据时，其值都和事务开始时候的内容是一致，禁止读取到别的事务未提交的数据（会造成幻读）。

SERIALIZABLE：序列化，代价最高最可靠的隔离级别，该隔离级别能防止脏读、不可重复读、幻读。

脏读 ：表示一个事务能够读取另一个事务中还未提交的数据。比如，某个事务尝试插入记录 A，此时该事务还未提交，然后另一个事务尝试读取到了记录 A。

不可重复读 ：是指在一个事务内，多次读同一数据。

幻读 ：指同一个事务内多次查询返回的结果集不一样。比如同一个事务 A 第一次查询时候有 n 条记录，但是第二次同等条件下查询却有 n+1 条记录，这就好像产生了幻觉。发生幻读的原因也是另外一个事务新增或者删除或者修改了第一个事务结果集里面的数据，同一个记录的数据内容被修改了，所有数据行的记录就变多或者变少了。

**174、说一下 mysql 常用的引擎？**

InnoDB 引擎：InnoDB 引擎提供了对数据库 acid 事务的支持，并且还提供了行级锁和外键的约束，它的设计的目标就是处理大数据容量的数据库系统。MySQL 运行的时候，InnoDB 会在内存中建立缓冲池，用于缓冲数据和索引。但是该引擎是不支持全文搜索，同时启动也比较的慢，它是不会保存表的行数的，所以当进行 select count(\*) from table 指令的时候，需要进行扫描全表。由于锁的粒度小，写操作是不会锁定全表的,所以在并发度较高的场景下使用会提升效率的。

MyIASM 引擎：MySQL 的默认引擎，但不提供事务的支持，也不支持行级锁和外键。因此当执行插入和更新语句时，即执行写操作的时候需要锁定这个表，所以会导致效率会降低。不过和 InnoDB 不同的是，MyIASM 引擎是保存了表的行数，于是当进行 select count(\*) from table 语句时，可以直接的读取已经保存的值而不需要进行扫描全表。所以，如果表的读操作远远多于写操作时，并且不需要事务的支持的，可以将 MyIASM 作为数据库引擎的首选。

**175、说一下 mysql 的行锁和表锁？**

MyISAM 只支持表锁，InnoDB 支持表锁和行锁，默认为行锁。

表级锁：开销小，加锁快，不会出现死锁。锁定粒度大，发生锁冲突的概率最高，并发量最低。

行级锁：开销大，加锁慢，会出现死锁。锁力度小，发生锁冲突的概率小，并发度最高。

**176、说一下乐观锁和悲观锁？**

乐观锁：每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在提交更新的时候会判断一下在此期间别人有没有去更新这个数据。

悲观锁：每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会阻止，直到这个锁被释放。

数据库的乐观锁需要自己实现，在表里面添加一个 version 字段，每次修改成功值加 1，这样每次修改的时候先对比一下，自己拥有的 version 和数据库现在的 version 是否一致，如果不一致就不修改，这样就实现了乐观锁。

**177、mysql 问题排查都有哪些手段？**

使用 show processlist 命令查看当前所有连接信息。

使用 explain 命令查询 SQL 语句执行计划。

开启慢查询日志，查看慢查询的 SQL。

**178、如何做 mysql 的性能优化？**

为搜索字段创建索引。

避免使用 select \*，列出需要查询的字段。

垂直分割分表。

选择正确的存储引擎。

**十八、Redis**

**179、redis 是什么？都有哪些使用场景？**

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API；

Redis使用场景：数据高并发的读写、海量数据的读写、对扩展性要求高的数据；

**180、redis 有哪些功能？**

数据缓存功能、分布式锁的功能、支持数据持久化、支持事务、支持消息队列

**181、redis 和 memecache 有什么区别？**

memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型；redis的速度比memcached快很多；redis可以持久化其数据；

**182、redis 为什么是单线程的？**

因为 cpu 不是 Redis 的瓶颈，Redis 的瓶颈最有可能是机器内存或者网络带宽。既然单线程容易实现，而且 cpu 又不会成为瓶颈，那就顺理成章地采用单线程的方案了。

关于 Redis 的性能，官方网站也有，普通笔记本轻松处理每秒几十万的请求。

而且单线程并不代表就慢 nginx 和 nodejs 也都是高性能单线程的代表。

**183、什么是缓存穿透？怎么解决？**

缓存穿透：指查询一个一定不存在的数据，由于缓存是不命中时需要从数据库查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，造成缓存穿透。

解决方案：最简单粗暴的方法如果一个查询返回的数据为空（不管是数据不存在，还是系统故障），我们就把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。

**184、redis 支持的数据类型有哪些？**

string、list、hash、set、zset

**185、redis 支持的 java 客户端都有哪些？**

Redisson、Jedis、lettuce等等，官方推荐使用Redisson

**186、jedis 和 redisson 有哪些区别？**

Jedis是Redis的Java实现的客户端，其API提供了比较全面的Redis命令的支持。

Redisson实现了分布式和可扩展的Java数据结构，和Jedis相比，功能较为简单，不支持字符串操作，不支持排序、事务、管道、分区等Redis特性。Redisson的宗旨是促进使用者对Redis的关注分离，从而让使用者能够将精力更集中地放在处理业务逻辑上

**187、怎么保证缓存和数据库数据的一致性？**

合理设置缓存的过期时间。

新增、更改、删除数据库操作时同步更新 Redis，可以使用事物机制来保证数据的一致性。

**188、redis 持久化有几种方式？**

Redis 的持久化有两种方式，或者说有两种策略：

RDB快照（Redis Database）：指定的时间间隔能对你的数据进行快照存储。

AOF日志（Append Only File）：每一个收到的写命令都通过write函数追加到文件中。

**189、redis 怎么实现分布式锁？**

Redis 分布式锁其实就是在系统里面占一个“坑”，其他程序也要占“坑”的时候，占用成功了就可以继续执行，失败了就只能放弃或稍后重试。

占坑一般使用 setnx(set if not exists)指令，只允许被一个程序占有，使用完调用 del 释放锁

**190、redis 分布式锁有什么缺陷？**

Redis 分布式锁不能解决超时的问题，分布式锁有一个超时时间，程序的执行如果超出了锁的超时时间就会出现问题。

**191、redis 如何做内存优化？**

尽可能使用散列表（hashes），散列表（是说散列表里面存储的数少）使用的内存非常小，所以你应该尽可能的将你的数据模型抽象到一个散列表里面。

比如你的web系统中有一个用户对象，不要为这个用户的名称，姓氏，邮箱，密码设置单独的key,而是应该把这个用户的所有信息存储到一张散列表里面。

**192、redis 淘汰策略有哪些？**

volatile-lru：从已设置过期时间的数据集（server. db[i]. expires）中挑选最近最少使用的数据淘汰。

volatile-ttl：从已设置过期时间的数据集（server. db[i]. expires）中挑选将要过期的数据淘汰。

volatile-random：从已设置过期时间的数据集（server. db[i]. expires）中任意选择数据淘汰。

allkeys-lru：从数据集（server. db[i]. dict）中挑选最近最少使用的数据淘汰。

allkeys-random：从数据集（server. db[i]. dict）中任意选择数据淘汰。

no-enviction（驱逐）：禁止驱逐数据。

**193、redis 常见的性能问题有哪些？该如何解决？**

主服务器写内存快照，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以主服务器最好不要写内存快照。  
Redis 主从复制的性能问题，为了主从复制的速度和连接的稳定性，主从库最好在同一个局域网内。

**十九、JVM**

**194.说一下 jvm 的主要组成部分？及其作用？**

类加载器（ClassLoader）、运行时数据区（Runtime Data Area）、执行引擎（Execution Engine）、本地库接口（Native Interface）

组件的作用： 首先通过类加载器（ClassLoader）会把 Java 代码转换成字节码，运行时数据区（Runtime Data Area）再把字节码加载到内存中，而字节码文件只是 JVM 的一套指令集规范，并不能直接交个底层操作系统去执行，因此需要特定的命令解析器执行引擎（Execution Engine），将字节码翻译成底层系统指令，再交由 CPU 去执行，而这个过程中需要调用其他语言的本地库接口（Native Interface）来实现整个程序的功能。

**195.说一下 jvm 运行时数据区？**

程序计数器、虚拟机栈、本地方法栈、堆、方法区

注：有的区域随着虚拟机进程的启动而存在，有的区域则依赖用户进程的启动和结束而创建和销毁。

**196.说一下堆栈的区别？**

栈内存存储的是局部变量而堆内存存储的是实体；栈内存的更新速度要快于堆内存，因为局部变量的生命周期很短；栈内存存放的变量生命周期一旦结束就会被释放，而堆内存存放的实体会被垃圾回收机制不定时的回收。

**197.队列和栈是什么？有什么区别？**

队列和栈都是被用来预存储数据的。

队列允许先进先出检索元素，但也有例外的情况，Deque 接口允许从两端检索元素。

栈和队列很相似，但它运行对元素进行后进先出进行检索。

**198.什么是双亲委派模型？**

双亲委派模型：如果一个类加载器收到了类加载的请求，它首先不会自己去加载这个类，而是把这个请求委派给父类加载器去完成，每一层的类加载器都是如此，这样所有的加载请求都会被传送到顶层的启动类加载器中，只有当父加载无法完成加载请求（它的搜索范围中没找到所需的类）时，子加载器才会尝试去加载类。

**199.说一下类加载的执行过程？**

类加载分为以下 5 个步骤：

加载：根据查找路径找到相应的 class 文件然后导入；

检查：检查加载的 class 文件的正确性；

准备：给类中的静态变量分配内存空间；

解析：虚拟机将常量池中的符号引用替换成直接引用的过程。符号引用就理解为一个标示，而在直接引用直接指向内存中的地址；

初始化：对静态变量和静态代码块执行初始化工作。

**200.** **怎么判断对象是否可以被回收（垃圾确定）？**

引用计数器算法：为每个对象创建一个引用计数，有对象引用时计数器 +1，引用被释放时计数 -1，当计数器为 0 时就可以被回收。它有一个缺点不能解决循环引用的问题，导致OOM发生；

可达性分析算法：将一系列根对象（GC Roots）作为起点，沿着这些起点向下搜索，搜索所走过的路径称为引用链。当一个对象到根对象没有任何引用链相连时，则证明此对象是可以被回收的。

**201.java 中都有哪些引用类型？**

强引用、软引用、弱引用、虚引用（幽灵引用/幻影引用）；

**202.说一下 jvm 有哪些垃圾回收算法？**

分代收集算法：将jvm堆内存分为不同的区域，再根据区域的特点采用不同的垃圾回收算法；一般分为新生代和老年代，新生代在再细分为Eden和Survivor From/To区（8:1:1），在新生代中，对象的存活率低，因此采用“复制”算法，将Edian区存活的对象复制到Survivor区；而老年代中对象存活率高且没有额外的中间做担保，因此采用“标记-清除”或者“标记-整理”算法来进行垃圾回收；

复制算法：主要是解决碎片问题，将内存一分为二，每次分配对象只用一半的内存空间，当一块内存空间用完时执行完MinorGC，将该块内存空间上存活的对象复制到另外一块内存上，然后将已使用过的内存空间一次清理掉；

· MinorGC 的过程（复制->清空->互换）：

· eden 、 servicorFrom 复制到 ServicorTo，年龄+1

· 清空 eden 、 servicorFrom

· ServicorTo 和 ServicorFrom 互换

标记-清除算法（Mark-Sweep）：

标记-整理算法（Mark-Compact）：

参考文献：https://www.jianshu.com/p/ea155fe22812

**203.说一下 jvm 有哪些垃圾回收器？**

Serial：最早的单线程串行垃圾回收器。

Serial Old：Serial 垃圾回收器的老年版本，同样也是单线程的，可以作为 CMS 垃圾回收器的备选预案。

ParNew：是 Serial 的多线程版本。

Parallel 和 ParNew 收集器类似是多线程的，但 Parallel 是吞吐量优先的收集器，可以牺牲等待时间换取系统的吞吐量。

Parallel Old 是 Parallel 老生代版本，Parallel 使用的是复制的内存回收算法，Parallel Old 使用的是标记-整理的内存回收算法。

CMS：一种以获得最短停顿时间为目标的收集器，非常适用 B/S 系统。

G1：一种兼顾吞吐量和停顿时间的 GC 实现，是 JDK 9 以后的默认 GC 选项。

**204.详细介绍一下 CMS 垃圾回收器？**

CMS 是英文 Concurrent Mark-Sweep 的简称，是以牺牲吞吐量为代价来获得最短回收停顿时间的垃圾回收器。对于要求服务器响应速度的应用上，这种垃圾回收器非常适合。在启动 JVM 的参数加上“-XX:+UseConcMarkSweepGC”来指定使用 CMS 垃圾回收器。

CMS 使用的是标记-清除的算法实现的，所以在 gc 的时候回产生大量的内存碎片，当剩余内存不能满足程序运行要求时，系统将会出现 Concurrent Mode Failure，临时 CMS 会采用 Serial Old 回收器进行垃圾清除，此时的性能将会被降低。

**205.新生代垃圾回收器和老生代垃圾回收器都有哪些？有什么区别？**

新生代回收器：Serial、ParNew、Parallel Scavenge

老年代回收器：Serial Old、Parallel Old、CMS

整堆回收器：G1

新生代垃圾回收器一般采用的是复制算法，复制算法的优点是效率高，缺点是内存利用率低；老年代回收器一般采用的是标记-整理的算法进行垃圾回收。

**206.简述分代垃圾回收器是怎么工作的？**

分代回收器有两个分区：老生代和新生代，新生代默认的空间占比总空间的 1/3，老生代的默认占比是 2/3。

新生代使用的是复制算法，新生代里有 3 个分区：Eden、To Survivor、From Survivor，它们的默认占比是 8:1:1，它的执行流程如下：

把 Eden + From Survivor 存活的对象放入 To Survivor 区；

清空 Eden 和 From Survivor 分区；

From Survivor 和 To Survivor 分区交换，From Survivor 变 To Survivor，To Survivor 变 From Survivor。

每次在 From Survivor 到 To Survivor 移动时都存活的对象，年龄就 +1，当年龄到达 15（默认配置是 15）时，升级为老生代。大对象也会直接进入老生代。

老生代当空间占用到达某个值之后就会触发全局垃圾收回，一般使用标记整理的执行算法。以上这些循环往复就构成了整个分代垃圾回收的整体执行流程。

**207.说一下 jvm 调优的工具？**

JDK 自带了很多监控工具，都位于 JDK 的 bin 目录下，其中最常用的是 jconsole 和 jvisualvm 这两款视图监控工具。

jconsole：用于对 JVM 中的内存、线程和类等进行监控；

jvisualvm：JDK 自带的全能分析工具，可以分析：内存快照、线程快照、程序死锁、监控内存的变化、gc 变化等。

**208.常用的 jvm 调优的参数都有哪些？**

-Xms2g：初始化推大小为 2g； -Xmx2g：堆最大内存为 2g；

-XX:+PrintGC：开启打印 gc 信息； -XX:+PrintGCDetails：打印 gc 详细信息。

-XX:NewRatio=4：设置年轻的和老年代的内存比例为 1:4；

-XX:SurvivorRatio=8：设置新生代 Eden 和 Survivor 比例为 8:2；

–XX:+UseParNewGC：指定使用 ParNew + Serial Old 垃圾回收器组合；

-XX:+UseParallelOldGC：指定使用 ParNew + ParNew Old 垃圾回收器组合；

-XX:+UseConcMarkSweepGC：指定使用 CMS + Serial Old 垃圾回收器组合；