## Java 分布式框架面试题合集

#### 1.什么是 ZooKeeper？

答：ZooKeeper 是一个开源的分布式应用程序协调服务，是一个典型的分布式数据一致性解决方案。设计目的是将那些复杂且容易出错的分布式一致性服务封装起来，构成一个高效可靠的系统，并以一系列简单易用的原子操作提供给用户使用。

#### 2.ZooKeeper 提供了哪些功能？

答：ZooKeeper 主要提供以下功能：

* 分布式服务注册与订阅：在分布式环境中，为了保证高可用性，通常同一个应用或同一个服务的提供方都会部署多份，达到对等服务。而消费者就须要在这些对等的服务器中选择一个来执行相关的业务逻辑，比较典型的服务注册与订阅，如 Dubbo。
* 分布式配置中心：发布与订阅模型，即所谓的配置中心，顾名思义就是发布者将数据发布到 ZooKeeper 节点上，供订阅者获取数据，实现配置信息的集中式管理和动态更新。
* 命名服务：在分布式系统中，通过命名服务客户端应用能够根据指定名字来获取资源、服务地址和提供者等信息。
* 分布式锁：这个主要得益于 ZooKeeper 为我们保证了数据的强一致性。

#### 3.ZooKeeper 有几种搭建模式？

答：ZooKeeper 通常有三种搭建模式：

* 单机模式：zoo.cfg 中只配置一个 server.id 就是单机模式了，此模式一般用在测试环境，如果当前主机宕机，那么所有依赖于当前 ZooKeeper 服务工作的其他服务器都不能进行正常工作；
* 伪分布式模式：在一台机器启动不同端口的 ZooKeeper，配置到 zoo.cfg 中，和单机模式相同，此模式一般用在测试环境；
* 分布式模式：多台机器各自配置 zoo.cfg 文件，将各自互相加入服务器列表，上面搭建的集群就是这种完全分布式。

#### 4.ZooKeeper 有哪些特性？

答： ZooKeeper 特性如下：

* 顺序一致性（Sequential Consistency）：来自相同客户端提交的事务，ZooKeeper 将严格按照其提交顺序依次执行；
* 原子性（Atomicity）：于 ZooKeeper 集群中提交事务，事务将“全部完成”或“全部未完成”，不存在“部分完成”；
* 单一系统镜像（Single System Image）：客户端连接到 ZooKeeper 集群的任意节点，其获得的数据视图都是相同的；
* 可靠性（Reliability）：事务一旦完成，其产生的状态变化将永久保留，直到其他事务进行覆盖；
* 实时性（Timeliness）：事务一旦完成，客户端将于限定的时间段内，获得最新的数据。

#### 5.以下关于 ZooKeeper 描述错误的是？

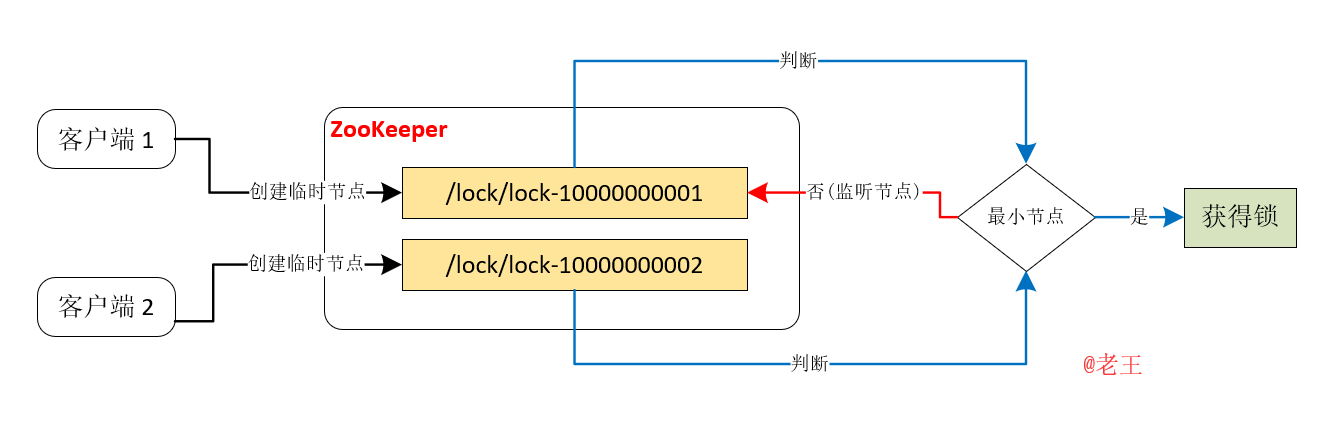
A：所有的节点都具有稳定的存储能力 B：ZooKeeper 任意节点之间都能够进行通信（消息发送 & 接收） C：为了提高性能，ZooKeeper 允许同一份数据存在一部分节点写成功，另一部分节点写失败 D：ZooKeeper 集群运行期间，只要半数以上节点存活，ZooKeeper 就能正常服务 答：C 题目解析：ZooKeeper 不允许同一份数据存在一部分节点写成功，另一部分节点写失败的情况，这不符合 ZooKeeper“一致性”的原则。

#### 6.ZooKeeper 如何实现分布式锁？

答：ZooKeeper 实现分布式锁的步骤如下：

* 客户端连接 ZooKeeper，并在 /lock 下创建临时的且有序的子节点，第一个客户端对应的子节点为 /lock/lock-10000000001，第二个为 /lock/lock-10000000002，以此类推。
* 客户端获取 /lock 下的子节点列表，判断自己创建的子节点是否为当前子节点列表中序号最小的子节点，如果是则认为获得锁，否则监听刚好在自己之前一位的子节点删除消息，获得子节点变更通知后重复此步骤直至获得锁；
* 执行业务代码；
* 完成业务流程后，删除对应的子节点释放锁。

整体流程如下图所示：



1

#### 7.ZooKeeper 如何实现分布式事务？

答：ZooKeeper 实现分布式事务，类似于两阶段提交，总共分为以下 4 步：

* 客户端先给 ZooKeeper 节点发送写请求；
* ZooKeeper 节点将写请求转发给 Leader 节点，Leader 广播给集群要求投票，等待确认；
* Leader 收到确认，统计投票，票数过半则提交事务；
* 事务提交成功后，ZooKeeper 节点告知客户端。

#### 8.集群中为什么要有主节点？

答：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，这就是主节点存在的意义。

#### 9.Dubbo 是什么？

答：Dubbo 是一款高性能、轻量级的开源 Java RPC 框架，它提供了三大核心能力：面向接口的远程方法调用，智能容错和负载均衡，以及服务自动注册和发现。

#### 10.Dubbo 有哪些特性？

答：Dubbo 特性如下：

* 面向接口代理的高性能 RPC 调用：提供高性能的基于代理的远程调用能力，服务以接口为粒度，为开发者屏蔽远程调用底层细节；
* 智能负载均衡：内置多种负载均衡策略，智能感知下游节点健康状况，显著减少调用延迟，提高系统吞吐量；
* 服务自动注册与发现：支持多种注册中心服务，服务实例上下线实时感知；
* 高度可扩展能力：遵循微内核+插件的设计原则，所有核心能力如 Protocol、Transport、Serialization 被设计为扩展点，平等对待内置实现和第三方实现；
* 运行期流量调度：内置条件、脚本等路由策略，通过配置不同的路由规则，轻松实现灰度发布，同机房优先等功能；
* 可视化的服务治理与运维：提供丰富服务治理、运维工具：随时查询服务元数据、服务健康状态及调用统计，实时下发路由策略、调整配置参数。

#### 11.Dubbo 有哪些核心组件？

答：Dubbo 核心组件如下：

* Provider：服务提供方
* Consumer：服务消费方
* Registry：服务注册与发现的注册中心
* Monitor：主要用来统计服务的调用次数和调用时间
* Container：服务的运行容器

#### 12.Dubbo 有哪些负载均衡策略？

答：Dubbo 负责均衡策略如下：

* 随机负载均衡（Random LoadBalance）：按权重设置随机概率，在一个截面上碰撞的概率高，但调用量越大分布越均匀，而且按概率使用权重后也比较均匀，有利于动态调整提供者权重；
* 轮询负载均衡（RoundRobin LoadBalance）：按公约后的权重设置轮询比率，存在慢的提供者累积请求的问题，比如：第二台机器很慢，但没挂，当请求调到第二台时就卡在那，久而久之，所有请求都卡在调到第二台上；
* 最少活跃调用数负载均衡（LeastActive LoadBalance）：使用最少活跃调用数，活跃数指调用前后计数差；
* 哈希负载均衡（ConsistentHash LoadBalance）：使用哈希值转发，相同参数的请求总是发到同一提供者。

**负载均衡配置如下** ：

服务端服务级别

<dubbo:service interface="xxx" loadbalance="roundrobin" />

客户端服务级别

<dubbo:reference interface="xxx" loadbalance="roundrobin" />

服务端方法级别

<dubbo:service interface="xxx">  
 <dubbo:method name="xxx" loadbalance="roundrobin"/>  
</dubbo:service>

客户端方法级别

<dubbo:reference interface="xxx">  
 <dubbo:method name="xxx" loadbalance="roundrobin"/>  
</dubbo:reference>

#### 13.Dubbo 不支持以下哪种协议？

A：dubbo://  
B：rmi://  
C：redis://  
D：restful://

答：D

题目解析：restful 一直编程规范，并不是一种传输协议，也不被 Dubbo 支持。

#### 14.Dubbo 默认使用什么注册中心，还有别的选择吗？

答：推荐使用 ZooKeeper 作为注册中心，还有 Nacos、Redis、Simple 注册中心（普通的 Dubbo 服务）。

#### 15.Dubbo 支持多注册中心吗？

答：Dubbo 支持同一服务向多注册中心同时注册，或者不同服务分别注册到不同的注册中心上去，甚至可以同时引用注册在不同注册中心上的同名服务。

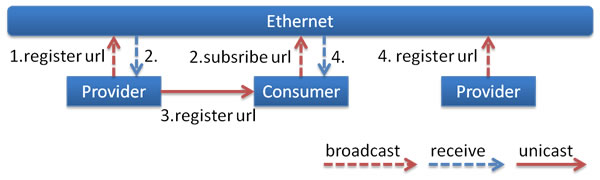
多注册中心注册：

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">  
 <dubbo:application name="world" />  
 <!-- 多注册中心配置 -->  
 <dubbo:registry id="hangzhouRegistry" address="10.20.141.150:9090" />  
 <dubbo:registry id="qingdaoRegistry" address="10.20.141.151:9010" default="false" />  
 <!-- 向多个注册中心注册 -->  
 <dubbo:service interface="com.alibaba.hello.api.HelloService" version="1.0.0" ref="helloService" registry="hangzhouRegistry,qingdaoRegistry" />  
</beans>

#### 16.Dubbo 支持的连接方式有哪些？

答：Dubbo 支持的主要连接方式有：组播、直连和 ZooKeeper 等注册中心。

**① 组播方式** ，不需要启动任何中心节点，只要广播地址一样，就可以互相发现。



2

1. 提供方启动时广播自己的地址
2. 消费方启动时广播订阅请求
3. 提供方收到订阅请求时，单播自己的地址给订阅者，如果设置了 unicast=false，则广播给订阅者
4. 消费方收到提供方地址时，连接该地址进行 RPC 调用

组播受网络结构限制，只适合小规模应用或开发阶段使用。组播地址段：224.0.0.0 ~ 239.255.255.255

配置

<dubbo:registry address="multicast://224.5.6.7:1234" />

或

<dubbo:registry protocol="multicast" address="224.5.6.7:1234" />

为了减少广播量，Dubbo 缺省使用单播发送提供者地址信息给消费者，如果一个机器上同时启了多个消费者进程，消费者需声明 unicast=false，否则只会有一个消费者能收到消息；当服务者和消费者运行在同一台机器上，消费者同样需要声明 unicast=false，否则消费者无法收到消息，导致 No provider available for the service 异常：

<dubbo:registry address="multicast://224.5.6.7:1234?unicast=false" />

或

<dubbo:registry protocol="multicast" address="224.5.6.7:1234">  
 <dubbo:parameter key="unicast" value="false" />  
</dubbo:registry>

**② 直连方式** ，注册中心本身就是一个普通的 Dubbo 服务，可以减少第三方依赖，使整体通讯方式一致。

<dubbo:registry protocol="zookeeper" address="N/A" file="./.dubbo-platform"/>

将 Simple 注册中心暴露成 Dubbo 服务：

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">  
 <!-- 当前应用信息配置 -->  
 <dubbo:application name="simple-registry" />  
 <!-- 暴露服务协议配置 -->  
 <dubbo:protocol port="9090" />  
 <!-- 暴露服务配置 -->  
 <dubbo:service interface="org.apache.dubbo.registry.RegistryService" ref="registryService" registry="N/A" ondisconnect="disconnect" callbacks="1000">  
 <dubbo:method name="subscribe"><dubbo:argument index="1" callback="true" /></dubbo:method>  
 <dubbo:method name="unsubscribe"><dubbo:argument index="1" callback="false" /></dubbo:method>  
 </dubbo:service>  
 <!-- 简单注册中心实现，可自行扩展实现集群和状态同步 -->  
 <bean id="registryService" class="org.apache.dubbo.registry.simple.SimpleRegistryService" />  
</beans>

引用 Simple Registry 服务：

<dubbo:registry address="127.0.0.1:9090" />

或者：

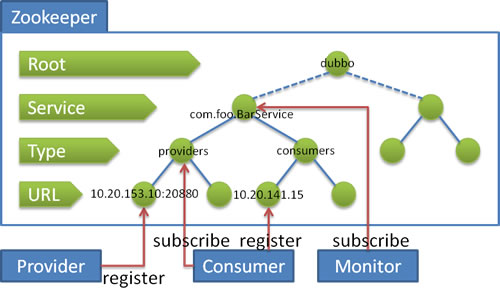
<dubbo:service interface="org.apache.dubbo.registry.RegistryService" group="simple" version="1.0.0" ... >

或者：

<dubbo:registry address="127.0.0.1:9090" group="simple" version="1.0.0" />

适用性说明：此 SimpleRegistryService 只是简单实现，不支持集群，可作为自定义注册中心的参考，但不适合直接用于生产环境。

**③ ZooKeeper 注册中心** ，Zookeeper 是 Apacahe Hadoop 的子项目，是一个树型的目录服务，支持变更推送，适合作为 Dubbo 服务的注册中心，工业强度较高，可用于生产环境，并推荐使用。



3

流程说明：

* 服务提供者启动时：向 /dubbo/com.foo.BarService/providers 目录下写入自己的 URL 地址
* 服务消费者启动时：订阅 /dubbo/com.foo.BarService/providers 目录下的提供者 URL 地址，并向 /dubbo/com.foo.BarService/consumers 目录下写入自己的 URL 地址
* 监控中心启动时: 订阅 /dubbo/com.foo.BarService 目录下的所有提供者和消费者 URL 地址

支持以下功能：

* 当提供者出现断电等异常停机时，注册中心能自动删除提供者信息
* 当注册中心重启时，能自动恢复注册数据，以及订阅请求
* 当会话过期时，能自动恢复注册数据，以及订阅请求
* 当设置 <dubbo:registry check="false" /> 时，记录失败注册和订阅请求，后台定时重试
* 可通过 <dubbo:registry username="admin" password="1234" /> 设置 zookeeper 登录信息
* 可通过 <dubbo:registry group="dubbo" /> 设置 zookeeper 的根节点，不设置将使用无根树
* 支持 \* 号通配符 <dubbo:reference group="\*" version="\*" />，可订阅服务的所有分组和所有版本的提供者

**Zookeeper 使用**

在 provider 和 consumer 中增加 zookeeper 客户端 jar 包依赖：

<dependency>  
 <groupId>org.apache.zookeeper</groupId>  
 <artifactId>zookeeper</artifactId>  
 <version>3.3.3</version>  
</dependency>

Dubbo 支持 zkclient 和 curator 两种 Zookeeper 客户端实现：

注意：在 2.7.x 的版本中已经移除了 zkclient 的实现，如果要使用 zkclient 客户端，需要自行拓展。

**使用 zkclient 客户端**

从 2.2.0 版本开始缺省为 zkclient 实现，以提升 zookeeper 客户端的健状性。[zkclient](https://github.com/sgroschupf/zkclient) 是 Datameer 开源的一个 Zookeeper 客户端实现。

缺省配置：

<dubbo:registry ... client="zkclient" />

或：

dubbo.registry.client=zkclient

或：

zookeeper://10.20.153.10:2181?client=zkclient

需依赖或直接[下载](http://repo1.maven.org/maven2/com/github/sgroschupf/zkclient)：

<dependency>  
 <groupId>com.github.sgroschupf</groupId>  
 <artifactId>zkclient</artifactId>  
 <version>0.1</version>  
</dependency>

**使用 curator 客户端**

从 2.3.0 版本开始支持可选 curator 实现。Curator 是 Netflix 开源的一个 Zookeeper 客户端实现。

如果需要改为 curator 实现，请配置：

<dubbo:registry ... client="curator" />

或：

dubbo.registry.client=curator

或：

zookeeper://10.20.153.10:2181?client=curator

需依赖或直接下载：

<dependency>  
 <groupId>com.netflix.curator</groupId>  
 <artifactId>curator-framework</artifactId>  
 <version>1.1.10</version>  
</dependency>

Zookeeper 单机配置:

<dubbo:registry address="zookeeper://10.20.153.10:2181" />

或：

<dubbo:registry protocol="zookeeper" address="10.20.153.10:2181" />

Zookeeper 集群配置：

<dubbo:registry address="zookeeper://10.20.153.10:2181?backup=10.20.153.11:2181,10.20.153.12:2181" />

或：

<dubbo:registry protocol="zookeeper" address="10.20.153.10:2181,10.20.153.11:2181,10.20.153.12:2181" />

同一 Zookeeper，分成多组注册中心：

<dubbo:registry id="chinaRegistry" protocol="zookeeper" address="10.20.153.10:2181" group="china" />  
<dubbo:registry id="intlRegistry" protocol="zookeeper" address="10.20.153.10:2181" group="intl" />

#### 17.什么是服务熔断？

答：在应用系统服务中，当依赖服务因访问压力过大而响应变慢或失败，上游服务为了保护系统整体的可用性，临时切断对下游服务的调用。这种牺牲局部，保全整体的措施就叫做熔断。

#### 18.Dubbo 可以对结果进行缓存吗？支持的缓存类型都有哪些？

答：可以，Dubbo 提供了声明式缓存，用于加速热门数据的访问速度，以减少用户加缓存的工作量。

Dubbo 支持的缓存类型有：

* lru 基于最近最少使用原则删除多余缓存，保持最热的数据被缓存；
* threadlocal 当前线程缓存，比如一个页面渲染，用到很多 portal，每个 portal 都要去查用户信息，通过线程缓存，可以减少这种多余访问；
* jcache 集成，可以桥接各种缓存实现。

配置如下：

<dubbo:reference interface="com.foo.BarService" cache="lru" />

或

<dubbo:reference interface="com.foo.BarService">  
 <dubbo:method name="findBar" cache="lru" />  
</dubbo:reference>

#### 19.Dubbo 有几种集群容错模式？

答：Dubbo 集群容错模式如下。

##### ① Failover Cluster

失败自动切换，当出现失败，重试其他服务器。通常用于读操作，但重试会带来更长延迟。可通过 retries="2" 来设置重试次数（不含第一次）。

重试次数配置如下：

<dubbo:service retries="2" />

或

<dubbo:reference retries="2" />

或

<dubbo:reference>  
 <dubbo:method name="findFoo" retries="2" />  
</dubbo:reference>

##### ② Failfast Cluster

快速失败，只发起一次调用，失败立即报错。通常用于非幂等性的写操作，比如新增记录。

##### ③ Failsafe Cluster

失败安全，出现异常时，直接忽略。通常用于写入审计日志等操作。

##### ④ Failback Cluster

失败自动恢复，后台记录失败请求，定时重发。通常用于消息通知操作。

##### ⑤ Forking Cluster

并行调用多个服务器，只要一个成功即返回。通常用于实时性要求较高的读操作，但需要浪费更多服务资源。可通过 forks=“2” 来设置最大并行数。

##### ⑥ Broadcast Cluster

广播调用所有提供者，逐个调用，任意一台报错则报错。通常用于通知所有提供者更新缓存或日志等本地资源信息。

更多java学习资料如下：

[Java高级架构师课程(总共122门课程，1460GB)](https://www.consultdog.com/index.php/front/44/)

[最强java面试视频课程(21门课程，126GB)](https://www.consultdog.com/index.php/front/78/)