https://deanwang1943.github.io/bugs/2018/10/05/面试/饿了么/dubbo 面试题/

# dubbo是什么

dubbo是一个分布式框架，远程服务调用的分布式框架，其核心部分包含：  
集群容错：提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持。  
远程通讯： 提供对多种基于长连接的NIO框架抽象封装，包括多种线程模型，序列化，以及“请求-响应”模式的信息交换方式。  
自动发现：基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。

# dubbo能做什么

透明化的远程方法调用，就像调用本地方法一样调用远程方法，只需简单配置，没有任何API侵入。

软负载均衡及容错机制，可在内网替代F5等硬件负载均衡器，降低成本，减少单点。

服务自动注册与发现，不再需要写死服务提供方地址，注册中心基于接口名查询服务提供者的IP地址，并且能够平滑添加或删除服务提供者。

1、默认使用的是什么通信框架，还有别的选择吗?

答：默认也推荐使用 netty 框架，还有 mina。

2、服务调用是阻塞的吗？

答：默认是阻塞的，可以异步调用，没有返回值的可以这么做。

# 一般使用什么注册中心？还有别的选择吗？

* Multicast注册中心： Multicast注册中心不需要任何中心节点，只要广播地址，就能进行服务注册和发现。基于网络中组播传输实现；
* Zookeeper注册中心： 基于分布式协调系统Zookeeper实现，采用Zookeeper的watch机制实现数据变更；
* redis注册中心： 基于redis实现，采用key/Map存储，住key存储服务名和类型，Map中key存储服务URL，value服务过期时间。基于redis的发布/订阅模式通知数据变更；
* Simple注册中心

ZooKeeper的节点是通过像树一样的结构来进行维护的，并且每一个节点通过路径来标示以及访问。除此之外，每一个节点还拥有自身的一些信息，包括：数据、数据长度、创建时间、修改时间等等。

# 默认使用什么序列化框架，你知道的还有哪些？

答：默认使用 Hessian 序列化，还有 Duddo、FastJson、Java 自带序列化。  
hessian是一个采用二进制格式传输的服务框架，相对传统soap web service，更轻量，更快速。

Hessian原理与协议简析：

http的协议约定了数据传输的方式，hessian也无法改变太多：

1) hessian中client与server的交互，基于http-post方式。

2) hessian将辅助信息，封装在http header中，比如“授权token”等，我们可以基于http-header来封装关于“安全校验”“meta数据”等。hessian提供了简单的”校验”机制。

3) 对于hessian的交互核心数据，比如“调用的方法”和参数列表信息，将通过post请求的body体直接发送，格式为字节流。

4) 对于hessian的server端响应数据，将在response中通过字节流的方式直接输出。

hessian的协议本身并不复杂，在此不再赘言；所谓协议(protocol)就是约束数据的格式，client按照协议将请求信息序列化成字节序列发送给server端，server端根据协议，将数据反序列化成“对象”，然后执行指定的方法，并将方法的返回值再次按照协议序列化成字节流，响应给client，client按照协议将字节流反序列话成”对象”。

# 服务提供者能实现失效踢出是什么原理？

答：服务失效踢出基于 zookeeper 的临时节点原理。

# 服务上线怎么不影响旧版本？

答：采用多版本开发，不影响旧版本。在配置中添加version来作为版本区分

# 如何解决服务调用链过长的问题？

答：可以结合 zipkin 实现分布式服务追踪。

# Dubbo的核心功能？

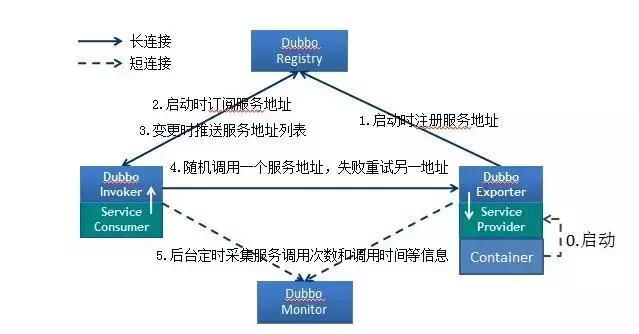
主要就是如下3个核心功能：

Remoting：网络通信框架，提供对多种NIO框架抽象封装，包括“同步转异步”和“请求-响应”模式的信息交换方式。

Cluster：服务框架，提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持。

Registry：服务注册，基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。

# Dubbo服务注册与发现的流程？

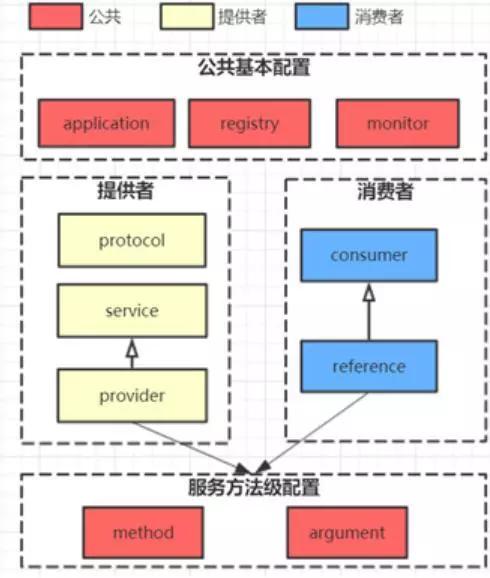


流程说明：

* Provider(提供者)绑定指定端口并启动服务
* 指供者连接注册中心，并发本机IP、端口、应用信息和提供服务信息发送至注册中心存储
* Consumer(消费者），连接注册中心 ，并发送应用信息、所求服务信息至注册中心
* 注册中心根据 消费 者所求服务信息匹配对应的提供者列表发送至Consumer 应用缓存。
* Consumer 在发起远程调用时基于缓存的消费者列表择其一发起调用。
* Provider 状态变更会实时通知注册中心、在由注册中心实时推送至Consumer

设计的原因：

* Consumer 与Provider 解偶，双方都可以横向增减节点数。
* 注册中心对本身可做对等集群，可动态增减节点，并且任意一台宕掉后，将自动切换到另一台
* 去中心化，双方不直接依懒注册中心，即使注册中心全部宕机短时间内也不会影响服务的调用
* 服务提供者无状态，任意一台宕掉后，不影响使用



# 说说核心的配置有哪些？

答：

核心配置有

dubbo:service/

dubbo:reference/

dubbo:protocol/

dubbo:registry/

dubbo:application/

dubbo:provider/

dubbo:consumer/

dubbo:method/

# dubbo 推荐用什么协议？

答：默认使用 dubbo 协议。

# 同一个服务多个注册的情况下可以直连某一个服务吗？

答：可以直连，修改配置即可，也可以通过 telnet 直接某个服务。

# dubbo 在安全机制方面如何解决的？

dubbo 通过 token 令牌防止用户绕过注册中心直连，然后在注册中心管理授权，dubbo 提供了黑白名单，控制服务所允许的调用方。

# 集群容错怎么做？

答：读操作建议使用 Failover 失败自动切换，默认重试两次其他服务器。写操作建议使用 Failfast 快速失败，发一次调用失败就立即报错。

# Dubbo的集群容错方案有哪些

* Failover Cluster。失败自动切换，当出现失败，重试其它服务器。通常用于读操作，但重试会带来更长延迟。
* 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。
* 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。
* 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。
* 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。
* 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。

# 在使用过程中都遇到了些什么问题？ 如何解决的？

1. 同时配置了 XML 和 properties 文件，则 properties 中的配置无效

只有 XML 没有配置时，properties 才生效。

2.dubbo 缺省会在启动时检查依赖是否可用，不可用就抛出异常，阻止 spring 初始化完成，check 属性默认为 true。

测试时有些服务不关心或者出现了循环依赖，将 check 设置为 false

3. 为了方便开发测试，线下有一个所有服务可用的注册中心，这时，如果有一个正在开发中的服务提供者注册，可能会影响消费者不能正常运行。

解决：让服务提供者开发方，只订阅服务，而不注册正在开发的服务，通过直连测试正在开发的服务。设置 dubbo:registry 标签的 register 属性为 false。

4.spring 2.x 初始化死锁问题。

在 spring 解析到 dubbo:service 时，就已经向外暴露了服务，而 spring 还在接着初始化其他 bean，如果这时有请求进来，并且服务的实现类里有调用 applicationContext.getBean() 的用法。getBean 线程和 spring 初始化线程的锁的顺序不一样，导致了线程死锁，不能提供服务，启动不了。

解决：不要在服务的实现类中使用 applicationContext.getBean(); 如果不想依赖配置顺序，可以将 dubbo:provider 的 deplay 属性设置为 - 1，使 dubbo 在容器初始化完成后再暴露服务。

5. 服务注册不上

检查 dubbo 的 jar 包有没有在 classpath 中，以及有没有重复的 jar 包

检查暴露服务的 spring 配置有没有加载

在服务提供者机器上测试与注册中心的网络是否通

6. 出现 RpcException: No provider available for remote service 异常

表示没有可用的服务提供者，

1). 检查连接的注册中心是否正确

2). 到注册中心查看相应的服务提供者是否存在

3). 检查服务提供者是否正常运行

7. 出现” 消息发送失败” 异常

通常是接口方法的传入传出参数未实现 Serializable 接口。

# dubbo 和 dubbox 之间的区别？

答：dubbox 是当当网基于 dubbo 上做了一些扩展，如加了服务可 restful 调用，更新了开源组件等。

# 你还了解别的分布式框架吗？

答：别的还有 spring 的 spring cloud，facebook 的 thrift，twitter 的 finagle 等。

# Dubbo 支持哪些协议，每种协议的应用场景，优缺点？

dubbo： 单一长连接和 NIO 异步通讯，适合大并发小数据量的服务调用，以及消费者远大于提供者。传输协议 TCP，异步，Hessian 序列化；

rmi： 采用 JDK 标准的 rmi 协议实现，传输参数和返回参数对象需要实现 Serializable 接口，使用 java 标准序列化机制，使用阻塞式短连接，传输数据包大小混合，消费者和提供者个数差不多，可传文件，传输协议 TCP。 多个短连接，TCP 协议传输，同步传输，适用常规的远程服务调用和 rmi 互操作。在依赖低版本的 Common-Collections 包，java 序列化存在安全漏洞；

webservice： 基于 WebService 的远程调用协议，集成 CXF 实现，提供和原生 WebService 的互操作。多个短连接，基于 HTTP 传输，同步传输，适用系统集成和跨语言调用；http： 基于 Http 表单提交的远程调用协议，使用 Spring 的 HttpInvoke 实现。多个短连接，传输协议 HTTP，传入参数大小混合，提供者个数多于消费者，需要给应用程序和浏览器 JS 调用；

hessian： 集成 Hessian 服务，基于 HTTP 通讯，采用 Servlet 暴露服务，Dubbo 内嵌 Jetty 作为服务器时默认实现，提供与 Hession 服务互操作。多个短连接，同步 HTTP 传输，Hessian 序列化，传入参数较大，提供者大于消费者，提供者压力较大，可传文件；

memcache： 基于 memcached 实现的 RPC 协议

redis： 基于 redis 实现的 RPC 协议

# Dubbo 集群的负载均衡有哪些策略

Dubbo 提供了常见的集群策略实现，并预扩展点予以自行实现。

Random LoadBalance: 随机选取提供者策略，有利于动态调整提供者权重。截面碰撞率高，调用次数越多，分布越均匀；

RoundRobin LoadBalance: 轮循选取提供者策略，平均分布，但是存在请求累积的问题；

LeastActive LoadBalance: 最少活跃调用策略，解决慢提供者接收更少的请求；

ConstantHash LoadBalance: 一致性 Hash 策略，使相同参数请求总是发到同一提供者，一台机器宕机，可以基于虚拟节点，分摊至其他提供者，避免引起提供者的剧烈变动；

# 服务调用超时问题怎么解决

dubbo在调用服务不成功时，默认是会重试两次的。这样在服务端的处理时间超过了设定的超时时间时，就会有重复请求，比如在发邮件时，可能就会发出多份重复邮件，执行注册请求时，就会插入多条重复的注册数据，那么怎么解决超时问题呢？如下

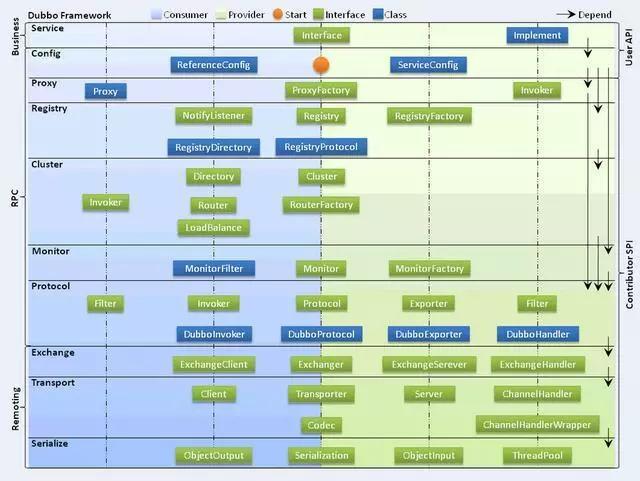
对于核心的服务中心，去除dubbo超时重试机制，并重新评估设置超时时间。

业务处理代码必须放在服务端，客户端只做参数验证和服务调用，不涉及业务流程处理  
全局配置实例

<dubbo:provider delay="-1" timeout="6000" retries="0"/>

当然Dubbo的重试机制其实是非常好的QOS保证，它的路由机制，是会帮你把超时的请求路由到其他机器上，而不是本机尝试，所以 dubbo的重试机器也能一定程度的保证服务的质量。但是请一定要综合线上的访问情况，给出综合的评估。

# Dubbo的架构设计？



Dubbo框架设计一共划分了10个层：

服务接口层（Service）：该层是与实际业务逻辑相关的，根据服务提供方和服务消费方的业务设计对应的接口和实现。

配置层（Config）：对外配置接口，以ServiceConfig和ReferenceConfig为中心。

服务代理层（Proxy）：服务接口透明代理，生成服务的客户端Stub和服务器端Skeleton。

服务注册层（Registry）：封装服务地址的注册与发现，以服务URL为中心。

集群层（Cluster）：封装多个提供者的路由及负载均衡，并桥接注册中心，以Invoker为中心。

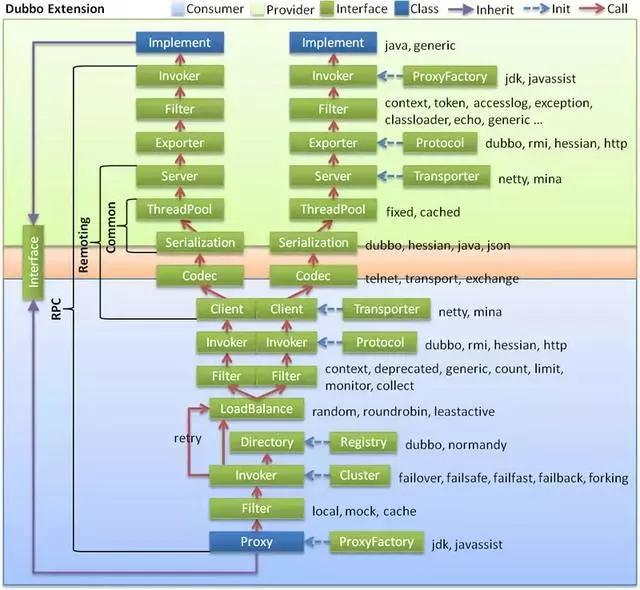
监控层（Monitor）：RPC调用次数和调用时间监控。

远程调用层（Protocol）：封将RPC调用，以Invocation和Result为中心，扩展接口为Protocol、Invoker和Exporter。

信息交换层（Exchange）：封装请求响应模式，同步转异步，以Request和Response为中心。

网络传输层（Transport）：抽象mina和netty为统一接口，以Message为中心。

# Dubbo的服务调用流程？



# Dubbo的服务治理？

* 过多的服务URL配置困难
* 负载均衡分配节点压力过大的情况下也需要部署集群
* 服务依赖混乱，启动顺序不清晰
* 过多服务导致性能指标分析难度较大，需要监控

# Dubbo的注册中心集群挂掉，发布者和订阅者之间还能通信么？

可以的，启动dubbo时，消费者会从zookeeper拉取注册的生产者的地址接口等数据，缓存在本地。

每次调用时，按照本地存储的地址进行调用

# Dubbo超时时间怎样设置

Dubbo超时时间设置有两种方式：

* 服务提供者端设置超时时间，在Dubbo的用户文档中，推荐如果能在服务端多配置就尽量多配置，因为服务提供者比消费者更清楚自己提供的服务特性。
* 服务消费者端设置超时时间，如果在消费者端设置了超时时间，以消费者端为主，即优先级更高。因为服务调用方设置超时时间控制性更灵活。如果消费方超时，服务端线程不会定制，会产生警告。

dubbo在调用服务不成功时，默认是会重试两次的。

# Dubbo和Spring Cloud的关系？

Dubbo

是 SOA 时代的产物，它的关注点主要在于服务的调用，流量分发、流量监控和熔断。而 Spring Cloud

诞生于微服务架构时代，考虑的是微服务治理的方方面面，另外由于依托了 Spirng、Spirng Boot

的优势之上，两个框架在开始目标就不一致，Dubbo 定位服务治理、Spirng Cloud 是一个生态。

# dubbo和spring cloud的区别？

最大的区别：Dubbo底层是使用Netty这样的NIO框架，是基于TCP协议传输的，配合以Hession序列化完成RPC通信。

而SpringCloud是基于Http协议+Rest接口调用远程过程的通信，相对来说，Http请求会有更大的报文，占的带宽也会更多。但是REST相比RPC更为灵活，服务提供方和调用方的依赖只依靠一纸契约，不存在代码级别的强依赖，这在强调快速演化的微服务环境下，显得更为合适，至于注重通信速度还是方便灵活性，具体情况具体考虑。