# dubbo

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。

RPC (Remote Procedure Call)即远程过程调用，传统意义上的 RPC 是指长连接数据交互。

## Dubbo常见问题

**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。

**3、一般使用什么注册中心？还有别的选择吗？**

答：推荐使用 zookeeper 注册中心，还有 Multicast注册中心, Redis注册中心, 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。

**4、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”。

**5、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。

**6、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

**7、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。

**8、说说核心的配置有哪些？**

核心配置有：

1) dubbo:service/

2) dubbo:reference/

3) dubbo:protocol/

4) dubbo:registry/

5) dubbo:application/

6) dubbo:provider/

7) dubbo:consumer/

8) dubbo:method/

**9、dubbo 推荐用什么协议？**

答：默认使用 dubbo 协议。

**10、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。

**11、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。

**12、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。

**13、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，

a. 检查连接的注册中心是否正确

b. 到注册中心查看相应的服务提供者是否存在

c. 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。

**14、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。

**15、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。

**16、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

**17、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；

**18、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

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。

为什么要用Dubbo？

内部使用了 Netty、Zookeeper，保证了高性能高可用性。使用 Dubbo 可以将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，可用于提高业务复用灵活扩展，使前端应用能更快速的响应多变的市场需求。

Dubbo 使用的是 RPC 通信，而 Spring Cloud 使用的是 HTTP RESTFul 方式。

dubbo都支持什么协议，推荐用哪种？

dubbo://（推荐）

rmi://

hessian://

http://

webservice://

thrift://

memcached://

redis://

rest://

Dubbo内置了哪几种服务容器？

Spring Container

Jetty Container

Log4j Container

Dubbo里面有哪几种节点角色？

Provider,Consumer,Registry,Monitor,Container

Dubbo有哪几种配置方式？

Spring 配置方式

Java API 配置方式

Dubbo 核心的配置有哪些？



在 Provider 上可以配置的 Consumer 端的属性有哪些？

1）timeout：方法调用超时

2）retries：失败重试次数，默认重试 2 次

3）loadbalance：负载均衡算法，默认随机

4）actives 消费者端，最大并发调用限制

Dubbo推荐使用什么序列化框架，你知道的还有哪些？

推荐使用Hessian序列化，还有Duddo、FastJson、Java自带序列化。

Dubbo默认使用的是什么通信框架，还有别的选择吗？

Dubbo 默认使用 Netty 框架，也是推荐的选择，另外内容还集成有Mina、Grizzly。

Dubbo有哪几种集群容错方案，默认是哪种？



Dubbo有哪几种负载均衡策略，默认是哪种？



当一个服务接口有多种实现时怎么做？

当一个接口有多种实现时，可以用 group 属性来分组，服务提供方和消费方都指定同一个 group 即可。

服务上线怎么兼容旧版本？

可以用版本号（version）过渡，多个不同版本的服务注册到注册中心，版本号不同的服务相互间不引用。这个和服务分组的概念有一点类似。

Dubbo可以对结果进行缓存吗？

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服务之间的调用是阻塞的吗？

默认是同步等待结果阻塞的，支持异步调用。

Dubbo 是基于 NIO 的非阻塞实现并行调用，客户端不需要启动多线程即可完成并行调用多个远程服务，相对多线程开销较小，异步调用会返回一个 Future 对象。

服务提供者能实现失效踢出是什么原理？

服务失效踢出基于 Zookeeper 的临时节点原理。

如何解决服务调用链过长的问题？

Dubbo 可以使用 Pinpoint 和 Apache Skywalking(Incubator) 实现分布式服务追踪，当然还有其他很多方案。

服务读写推荐的容错策略是怎样的？

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必须依赖的包有哪些？

Dubbo 必须依赖 JDK，其他为可选。

Dubbo的管理控制台能做什么？

管理控制台主要包含：路由规则，动态配置，服务降级，访问控制，权重调整，负载均衡，等管理功能。

说说 Dubbo 服务暴露的过程。

Dubbo 会在 Spring 实例化完 bean 之后，在刷新容器最后一步发布 ContextRefreshEvent 事件的时候，通知实现了 ApplicationListener 的 ServiceBean 类进行回调 onApplicationEvent 事件方法，Dubbo 会在这个方法中调用 ServiceBean 父类 ServiceConfig 的 export 方法，而该方法真正实现了服务的（异步或者非异步）发布。

在使用过程中都遇到了些什么问题？

Dubbo 的设计目的是为了满足高并发小数据量的 rpc 调用，在大数据量下的性能表现并不好，建议使用 rmi 或 http 协议。

## Dubbo的总体架构

**Double服务框架其核心部分：**

1.远程通讯：提供对多种基于长连接的NIO框架抽象封装，包括多线程模型，序列化，以及“请求-响应”模式的信息交换方式

2.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

3.自动发现，基于注册中心目录服务，是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继承了当前主流的网络通信框架，主要包括如下几个：Mina,Netty,Grizzly

Dubbo支持多种协议，如下所示：Dubbo协议 Hessian协议 HTTP协议 RMI协议 WebService协议 Thrift协议 Memcached协议 Redis协议

**Dubbo底层通信是通过支持异步、事件驱动的NIO网络编程框架**，如：Netty、Mina、Grizzly，此框架是典型的Reactor模式使用，使得单个线程处理多个请求，且支持多请求并行执行，NIO接受请求处理流程是读取请求数据解码执行业务逻辑编码发送回应消息，Dubbo是对NIO框架的再次抽象封装，加入一些Dubbo需要的逻辑，通过抽象扩展Handler完成，如HeaderExchangerHandler完成请求-响应模式、同步转异步模式消息发送，AllChannelHandler通过线程池完成请求、响应、连接等并行执行，下面会详细介绍。

**Dubbo超时时间怎样设置？**

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通讯协议：

dubbo

Dubbo 缺省协议采用单一长连接和 NIO 异步通讯，适合于小数据量大并发的服务调用，以及服务消费者机器数远大于服务提供者机器数的情况。

rmi://

RMI 协议采用 JDK 标准的 java.rmi.\* 实现，采用阻塞式短连接和 JDK 标准序列化方式。注意：如果正在使用 RMI 提供服务给外部访问 1，同时应用里依赖了老的 common-collections 包 2的情况下，存在反序列化安全风险

服务调用默认是阻塞的，可以异步调用，没有返回值的可以这么做。

默认使用Hessian序列化，还有Duddo、FastJson、Java自带序列化。

服务提供者能实现失效踢出是什么原理？

服务失效踢出基于zookeeper的临时节点原理

如何解决服务调用链过长的问题？:可以结合zipkin实现分布式服务追踪。

核心配置有 dubbo:service/ dubbo:reference/ dubbo:protocol/ dubbo:registry/ dubbo:application/ dubbo:provider/ dubbo:consumer/ dubbo:method/

**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，可以直接new配置类，也可以通过spring解析配置生成配置类。

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，以ServiceProxy为中心，扩展接口为ProxyFactory。

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，扩展接口为RegistryFactory、Registry和RegistryService。可能没有服务注册中心，此时服务提供方直接暴露服务。

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，扩展接口为Cluster、Directory、Router和LoadBalance。将多个服务提供方组合为一个服务提供方，实现对服务消费方来透明，只需要与一个服务提供方进行交互。

监控层（Monitor）：RPC调用次数和调用时间监控，以Statistics为中心，扩展接口为MonitorFactory、Monitor和MonitorService。

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。Protocol是服务域，它是Invoker暴露和引用的主功能入口，它负责Invoker的生命周期管理。Invoker是实体域，它是Dubbo的核心模型，其它模型都向它靠扰，或转换成它，它代表一个可执行体，可向它发起invoke调用，它有可能是一个本地的实现，也可能是一个远程的实现，也可能一个集群实现。

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，扩展接口为Exchanger、ExchangeChannel、ExchangeClient和ExchangeServer。

网络传输层（Transport）：抽象mina和netty为统一接口，以Message为中心，扩展接口为Channel、Transporter、Client、Server和Codec。

数据序列化层（Serialize）：可复用的一些工具，扩展接口为Serialization、 ObjectInput、ObjectOutput和ThreadPool。

**Dubbo集群健壮性：**

监控中心宕掉不影响使用，只是丢失部分采样数据

数据库宕掉后，注册中心仍能通过缓存提供服务列表查询，但不能注册新服务

注册中心对等集群，任意一台宕掉后，将自动切换到另一台

注册中心全部宕掉后，服务提供者和服务消费者仍能通过本地缓存通讯

服务提供者无状态，任意一台宕掉后，不影响使用。

服务提供者全部宕掉后，服务消费者应用无法使用，并无限次重连等待服务提供者恢复

## Dubbo原理

服务层：

Dubbo最大的特点就是按照分层的方式来架构，使用这种方式可以实现各个层之间最大限度地松耦合。Dubbo的框架设计一共划分为10层，有服务接口层，配置层，服务代理层，集群层，监控层，服务注册层，远程调用层，信息交换层，网络传输层，数据序列化层。其中我觉得最主要的就是服务代理层，服务注册层与集群层，服务代理层会将根据你写的接口生成一个代理类，用于处理请求，而服务注册层则会将服务地址暴露给注册中心，请求过来后再由底层转发给各个代理实现类去处理，并响应。

测试和生产公用一套zookeeper，怎么保证消费不冲突

1.dubbo白名单（Filter过滤器）

首先实现com.alibaba.dubbo.rpc.Filter接口：并在实现的接口中根据服务的ip过滤掉不想用的服务。

其次在resources目录下添加纯文本文件META-INF/dubbo/com.alibaba.dubbo.rpc.Filter，内容如下： xxxFilter=com.xxx.AuthorityFilter

并且修改dubbo的provider配置文件，在dubbo:provider中添加配置的filter， 内容如下：<dubbo:provider filter="xxxFilter" />

2.服务分组

<!--服务-->

<dubbo:service group="feedback" interface="com.xxx.IndexService" />

<dubbo:service group="member" interface="com.xxx.IndexService" />

<!--引用-->

<dubbo:reference id="feedbackIndexService" group="feedback" interface="com.xxx.IndexService" />

<dubbo:reference id="memberIndexService" group="member" interface="com.xxx.IndexService" />

3. 多版本

<dubbo:service interface="com.foo.BarService" version="1.0.0" />

## 分布式事务

1使用spring aop配置事务

2 使用zookeeper或redis获取分布式事务锁

3 使用定时任务下单

4 使用消息队列完成最终一致性

## 分布式集群下如何做到唯一序列号

1. 数据库自增长序列或字段:每个Master库设置的起始数字不一样，步长一样，可以是Master的个数。比如：Master1 生成的是 1，4，7，10，Master2生成的是2,5,8,11 Master3生成的是 3,6,9,12。

2. UUID

3. UUID的变种:

1）为了解决UUID不可读，可以使用UUID to Int64的方法。

2）为了解决UUID无序的问题，NHibernate在其主键生成方式中提供了Comb算法（combined guid/timestamp）。保留GUID的10个字节，用另6个字节表示GUID生成的时间（DateTime）。

4. Redis生成ID

5. Twitter的snowflake算法

6. 利用zookeeper生成唯一ID

分布式锁目前比较常用的几种方案：

基于数据库实现分布式锁

基于缓存（redis , memcached）实现分布式锁

基于zookeeper实现分布式锁

## 分布式系统优化

**分布式系统优化原则：**

减少跨机网络交互

尽可能带sharding key

分页优化

减少数据读写热点

切分颗粒度尽可能细

减少锁开销

尽可能规避分布式事务

尽一切可能利用单机资源

单机事务

单机join

好的存储模型，就是尽可能多多做到以下几点：

尽可能走内存

尽可能将一次要查询到到数据物理到放在一起

通过合理到数据冗余，减少走网络到次数

合理并行提升响应时间

读取数据瓶颈，可以通过加slave节点解决

写入瓶颈，用规则sharding和扩容来解决

全部信息索引：

http://weibo.com/1765738567/C72pRgl83

一致性选择

无主机方案

有主机方案

分布式存储

key-value多机扩展

CAP和分布式系统多一致性

## 服务治理

服务治理的核心是：服务性能KPI统计、告警、服务健康管理、灵活的弹性伸缩策略、故障自动迁移、服务限流和服务降级等多种治理手段，保障服务高质量运行

常见的服务治理：服务降级、服务流控、服务动态扩展、超时控制、优先级调度、负载均衡策略调整、分组调整、等

## 高可用系统常用解决手段

1. 系统拆分，读写分离
2. 降级：当故障发生后，怎么减小故障所造成的损失，让系统服务能力不直接降到0

2.1限流

2.2异步调用：弱依赖调用通过消息中间件的方式来实现。

2.3同步调用设置超时时间

2.4失败重试

如何实现负载均衡，有哪些算法可以实现

数据的垂直拆分水平拆分

zookeeper原理及适用场景

zookeeper watch机制

那么要实现Watch，就必须实现org.apache.zookeeper.Watcher接口，并且将实现类的对象传入到可以Watch的方法中。Zookeeper中所有读操作（getData()，getChildren()，exists()）都可以设置Watch选项。Watch事件具有one-time trigger（一次性触发）的特性，如果Watch监视的Znode有变化，那么就会通知设置该Watch的客户端。

## redis/zk节点宕机如何处理?

## 其它

分布式事务是指事务的参与者，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。