# dubbo

Dubbo是基于远程服务调用的分布式框架，其核心部分包含：集群容错，多协议支持，软负载均衡，地址路由，动态配置等集群支持。

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默认使用netty作为通信框架，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。

Dubbo默认使用 zookeeper 注册中心，还有 Multicast注册中心, Redis注册中心, Simple注册中心。通过创建和监听znode节点来实现服务注册和发现。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。

Dubbo默认使用 dubbo 协议,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。

## 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。

## 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/

## 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。

## 在使用过程中都遇到了些什么问题？ 如何解决的？

比较常遇到就是因为服务端操作比较繁琐，造成超时的问题，之前做这个系统时因为有个对账模块因为数据量大偶尔会这样，这个要么优化代码逻辑，要么配置长一点的超时时间。

另一个就是因为传递的对象没有实现序列化接口而报错，印象比较深刻的就是list的sublist方法返回的list是未实现序列化的，这个要么换其它数据格式，要么转换成实现了序列化的集合。

其它的比如配置未生效，启动抱服务不可用，服务注册不上等等，一般都是环境或配置的问题，根据异常提示仔细检查一下就可以了。

## 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。

## 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，多个短连接，TCP 协议传输，同步传输，一般使用在大数据或文件传递方面。java 序列化存在安全漏洞；

webservice:基于 WebService 的远程调用协议，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

thrift:

## 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

## 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。

## 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。

## 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框架

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。

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。

## 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" />