Dubbo 是什么？

Dubbo 是一个分布式服务框架，致力于提供高性能和透明化的 RPC 远程服务调用方案，以及 SOA 服务治理方案。

简单的说，dubbo 就是个服务框架，如果没有分布式的需求，其实是不需要用的，只有在分布式的时候，才有 dubbo 这样的分布式服务框架的需求，并且本质上是个服务调用的东东，说白了就是个远程服务调用的分布式框架（告别 Web Service 模式中的 WSdl，以服务者与消费者的方式在 dubbo 上注册）。

其核心部分包含:

1. 远程通讯:

提供对多种基于长连接的 NIO 框架抽象封装，包括多种线程模型，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，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？

1.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。

2.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。

3. 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。

Dubbo 采用全 spring 配置方式，透明化接入应用，对应用没有任何 API 侵入，只需用 Spring 加载 Dubbo 的配置即可，Dubbo 基于 Spring 的 Schema 扩展进行加载。

Dubbo 的架构和设计思路

Dubbo 框架具有极高的扩展性，主要采用微核+插件体系，并且文档齐全，很方便二次开发，适应性极强。

Dubbo 的总体架构，如图所示：

Dubbo 框架设计一共划分了 10 个层，而最上面的 Service 层是留给实际想要使用 Dubbo 开发分布式服务的开发者实现业务逻辑的接口层。图中左边淡蓝背景的为服务消费方使用的接口，右边淡绿色背景的为服务提供方使用的接口，位于中轴线上的为双方都用到的接口。

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，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。

和淘宝 HSF 相比，Dubbo 的特点是什么？

1. Dubbo 比 HSF 的部署方式更轻量

HSF 要求使用指定的 JBoss 等容器，还需要在 JBoss 等容器中加入 sar 包扩展，对用户运行环境的侵入性大，如果你要运行在 Weblogic 或 Websphere 等其它容器上，需要自行扩展容器以兼容 HSF 的 ClassLoader 加载，而 Dubbo 没有任何要求，可运行在任何 Java 环境中。

2. Dubbo 比 HSF 的扩展性更好，很方便二次开发

一个框架不可能覆盖所有需求，Dubbo 始终保持平等对待第三方理念，即所有功能，都可以在不修改 Dubbo 原生代码的情况下，在外围扩展，包括 Dubbo 自己内置的功能，也和第三方一样，是通过扩展的方式实现的，而 HSF 如果你要加功能或替换某部分实现是很困难的，比如支付宝和淘宝用的就是不同的 HSF 分支，因为加功能时改了核心代码，不得不拷一个分支单独发展，HSF 现阶段就算开源出来，也很难复用，除非对架构重写。

3. HSF 依赖比较多内部系统

比如配置中心，通知中心，监控中心，单点登录等等，如果要开源还需要做很多剥离工作，而 Dubbo 为每个系统的集成都留出了扩展点，并已梳理干清所有依赖，同时为开源社区提供了替代方案，用户可以直接使用。

4. Dubbo 比 HSF 的功能更多

除了 ClassLoader 隔离，Dubbo 基本上是 HSF 的超集，Dubbo 也支持更多协议，更多注册中心的集成，以适应更多的网站架构。

Dubbo 适用于哪些场景？

1.RPC 分布式服务

当网站变大后，不可避免的需要拆分应用进行服务化，以提高开发效率，调优性能，节省关键竞争资源等。

比如：为了适用不断变化的市场需求，以及多个垂直应用之间数据交互方便，我们把公共的业务抽取出来作为独立的模块，为其他的应用提供服务，系统逐渐依赖于抽象和 rpc 远程服务调用。

2.配置管理

当服务越来越多时，服务的 URL 地址信息就会爆炸式增长，配置管理变得非常困难，F5 硬件负载均衡器的单点压力也越来越大。

3.服务依赖

当进一步发展，服务间依赖关系变得错踪复杂，甚至分不清哪个应用要在哪个应用之前启动，架构师都不能完整的描述应用的架构关系。

4.服务扩容

接着，服务的调用量越来越大，服务的容量问题就暴露出来，这个服务需要多少机器支撑？什么时候该加机器？等等.....

在遇到这些问题时，都可以用 *Dubbo* 来解决。

2020 年最新 Java 架构师系统进阶资料免费领取

需要【 一线大厂最新面试题与答案汇总 】的朋友请加 QQ 群/ 微信群 分布式/源码/性能交流 QQ 群：833977986

微信扫描二维码获取资料学习

【 一线大厂最新面试题与答案汇总 】 包含阿里，京东、百 度、腾讯、等一线大厂最新面试题与面试题答案。群里还会 讨论 Kafka、Mysql、Tomcat、Docker、Spring、MyBatis、 Nginx、Netty、Dubbo、Redis、Netty、Spring cloud、 JVM、分布式、高并发、性能调优、微服务等架构师最新技能 与问题学习——进群备注好信息即可免费领取。