链滴

Dubbo架构设计解析

作者：s3153478287

原文链接：https://ld246.com/article/1461564970264

来源网站：链滴

总体架构

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。

下面，结合Dubbo官方文档，我们分别理解一下框架分层架构中，各个层次的设计要点：

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Service roxy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RegistryFact ry、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MonitorFacto y、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、Inv ker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Channe 、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、ObjectO tput和ThreadPool。

从上图可以看出，Dubbo对于服务提供方和服务消费方，从框架的10层中分别提供了各自需要关心扩展的接口，构建整个服务生态系统（服务提供方和服务消费方本身就是一个以服务为中心的）。

根据官方提供的，对于上述各层之间关系的描述，如下所示：

在RPC中，Protocol是核心层，也就是只要有Protocol + Invoker + Exporter就可以完成非透明的RP 调用，然后在Invoker的主过程上Filter拦截点。

图中的Consumer和Provider是抽象概念，只是想让看图者更直观的了解哪些类分属于客户端与服务端，不用Client和Server的原因是Dubbo在很多场景下都使用Provider、Consumer、Registry、Mon tor划分逻辑拓普节点，保持统一概念。

而Cluster是外围概念，所以Cluster的目的是将多个Invoker伪装成一个Invoker，这样其它人只要关 Protocol层Invoker即可，加上Cluster或者去掉Cluster对其它层都不会造成影响，因为只有一个提供

时，是不需要Cluster的。

Proxy层封装了所有接口的透明化代理，而在其它层都以Invoker为中心，只有到了暴露给用户使用时才用Proxy将Invoker转成接口，或将接口实现转成Invoker，也就是去掉Proxy层RPC是可以Run的，是不那么透明，不那么看起来像调本地服务一样调远程服务。

而Remoting实现是Dubbo协议的实现，如果你选择RMI协议，整个Remoting都不会用上，Remotin内部再划为Transport传输层和Exchange信息交换层，Transport层只负责单向消息传输，是对Mina Netty、Grizzly的抽象，它也可以扩展UDP传输，而Exchange层是在传输层之上封装了Request-Resonse语义。

Registry和Monitor实际上不算一层，而是一个独立的节点，只是为了全局概览，用层的方式画在一。

从上面的架构图中，我们可以了解到，Dubbo作为一个分布式服务框架，主要具有如下几个核心的要：