# Dubbo:的使用

SOA的使用：服务框架

**二、Dubbo是什么**

Dubbo 是阿里巴巴公司开源的一个高性能优秀的服务框架，使得应用可通过高性能的 RPC 实现服务的输出和输入功能，以及SOA服务治理方案。

（1）主要核心部件：  
Remoting: 网络通信框架，实现了 sync-over-async 和 request-response 消息机制.  
RPC: 一个远程过程调用的抽象，支持负载均衡、容灾和集群功能  
Registry: 服务目录框架用于服务的注册和服务事件发布和订阅

（2）几点我的理解

Dubbo使用Hessian协议实现，这里的高性能的 RPC指的就是Hessian协；

Dubbo是一个远程服务调用在分布式系统中的一个实现框架，不再使用以前的Web service方式，而是通过服务提供者和消费者的方式调用；

并且通过在注册中心注册，消费者无需知道提供方的地址，可以通过注册中心读取，注册中心作为中间层，在中间层又可以实现负载均衡等，

这样就不需要负载均衡硬件，真正的实现大规模分布式系统的远程服务调用；

同时在注册中心宕机的情况下，支持服务提供者和消费者直接通过地址调用，在容错上表现较好；

并且改变服务提供者不需要通知服务消费者，实现了平滑删除和添加；

**三、Dubbo解决了哪些问题**

* 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。

**四、Dubbo的设计结构和工作原理**

（1）设计结构



Provider：  
暴露服务方称之为“服务提供者”。  
Consumer：  
调用远程服务方称之为“服务消费者”。  
Registry：  
服务注册与发现的中心目录服务称之为“服务注册中心”。  
Monitor：

统计服务的调用次调和调用时间的日志服务称之为“服务监控中心”。

Container:  
服务运行容器。

（2）调用过程

1. 服务容器负责启动，加载，运行服务提供者。
2. 服务提供者在启动时，向注册中心注册自己提供的服务。
3. 服务消费者在启动时，向注册中心订阅自己所需的服务。
4. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。
5. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。
6. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

（3）Dubbo的特性

连通性：

* 注册中心负责服务地址的注册与查找，相当于目录服务，服务提供者和消费者只在启动时与注册中心交互，注册中心不转发请求，压力较小
* 监控中心负责统计各服务调用次数，调用时间等，统计先在内存汇总后每分钟一次发送到监控中心服务器，并以报表展示
* 服务提供者向注册中心注册其提供的服务，并汇报调用时间到监控中心，此时间不包含网络开销
* 服务消费者向注册中心获取服务提供者地址列表，并根据负载算法直接调用提供者，同时汇报调用时间到监控中心，此时间包含网络开销
* 注册中心，服务提供者，服务消费者三者之间均为长连接，监控中心除外
* 注册中心通过长连接感知服务提供者的存在，服务提供者宕机，注册中心将立即推送事件通知消费者
* 注册中心和监控中心全部宕机，不影响已运行的提供者和消费者，消费者在本地缓存了提供者列表
* 注册中心和监控中心都是可选的，服务消费者可以直连服务提供者

健状性：

* 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，服务消费者应用将无法使用，并无限次重连等待服务提供者恢复

伸缩性：

* 注册中心为对等集群，可动态增加机器部署实例，所有客户端将自动发现新的注册中心
* 服务提供者无状态，可动态增加机器部署实例，注册中心将推送新的服务提供者信息给消费者

**五、Dubbo的集群容错机制**

当服务调用失败时（比如响应超时），根据我们的业务不同，可以使用不同的策略来应对这种失败。

比如我们调用的服务是一个查询服务，不会修改数据库，那么可以给该服务设置容错方式为failover ， 当调用失败时，自动切换到其他服务提供者去调用，当失败次数超过指定重试次数，那么就抛出错误；  
如果服务是更新数据的服务，那就不能使用失败重试的方式了， 因为这样可能产生数据重复修改的问题，比如调用提供者A的插入用户方法，但是该方法业务逻辑复杂，执行过程很慢，导致响应超时， 那么此时如果再去调用另外一个服务提供者的插入用户方法，将会又重复插入同一个用户。 对于这种类型的服务，可以使用容错方式为failfast，如果第一次调用失败，立即报错，不需要重试；

另外还有下面几种容错类型：  
failsafe 出现错误，直接忽略，不重试也不报错  
failback 失败后不报错，会将该失败请求，定时重发，适合消息通知类型的服务  
forking 并行调用多个服务器，只要在某一台提供者上面成功，那么方法返回， 适合实时性要求较高的查询服务， 但是要牺牲性能。因为每台服务器会做同一个操作  
broadcast 广播调用所有服务提供者，逐个调用，任意一台报错则报错。 适合与更新每台提供者上面的缓存这种类型的服务。

**六、Dubbo使用的多协议**

dubbo提供了多种协议给用户选择， 如dubbo、hessian、rmi 。 并可为每个服务指定不同的传输协议，粒度可以细化到方法， 不同服务在性能上适用不同协议进行传输，比如大数据用短连接协议，小数据大并发用长连接协议。

**七、可以替代Dubbo的组件**

Hessian、spring httpinvoke等。

**八、Dubbo的特性**

相比其他同类组件，Dubbo有自己的一些优势：

（1）服务注册中心  
相比Hessian类RPC框架，Dubbo有自己的服务中心， 写好的服务可以注册到服务中心， 客户端从服务中心寻找服务，然后再到相应的服务提供者机器获取服务  
通过服务中心可以实现集群、负载均衡、高可用(容错) 等重要功能。

服务中心一般使用zookeeper实现， 也有redis和其他一些方式 。 以使用zookeeper作为服务中心为例， 服务提供者启动后会在zookeeper的 /dubbo节点下创建提供的服务节点，包含服务提供者ip、port等信息。 服务提供者关闭时会从zookeeper中移除对应的服务。

服务使用者会从注册中心zookeeper中寻找服务，同一个服务可能会有多个提供者， Dubbo会帮我们找到合适的服务提供者，也就是针对服务提供者的负载均衡。

（2）负载均衡  
当同一个服务有多个提供者在提供服务时， 客户端如何正确的选择提供者实现负载均衡dubbo也给我们提供了几种方案：  
random 随机选提供者，并可以给提供者设置权重  
roundrobin 轮询选择提供者  
leastactive 最少活跃调用数，相同活跃数的随机，活跃数指调用前后计数差。使慢的提供者收到更少请求，因为越慢的提供者的调用前后计数差会越大。  
consistenthash 一致性hash，相同参数的请求发到同一台机器上

（3）简化测试，允许直连提供者  
在开发阶段为了方便测试，通常系统客户端能指定调用某个服务提供者，那么可以在引用服务时加一个url参数去指定服务提供者

|  |  |
| --- | --- |
| 1  2 | <dubbo:reference  id="xxxService"interface="com.alibaba.xxx.XxxService"url="dubbo://localhost:20890"/> |

（4）服务版本，服务分组  
在Dubbo配置文件中可以通过制定版本实现连接制定提供者，  
也就是通过服务版本可以控制服务的不兼容升级；  
当同一个服务有多种实现时，可以使用服务分组进行区分。