# dubbo

1. dubbo产生背景

不同系统之间方法的调用, 之前通过RMI或Hessian, 通过配置服务的URL地址进行调用，通过 F5 等硬件进行负载均衡。**当服务越来越多时，服务 URL 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.**

服务注册中心，动态的注册和发现服务，使服务的位置透明。并通过在消费方获取服务提供方地址列表，实现软负载均衡和 Failover，降低对 F5 硬件负载均衡器的依赖.

统计服务调用量, 响应时间, 动态调整权重等.

1. 架构.

| **节点** | **角色说明** |
| --- | --- |
| Provider | 暴露服务的服务提供方 |
| Consumer | 调用远程服务的服务消费方 |
| Registry | 服务注册与发现的注册中心 |
| Monitor | 统计服务的调用次数和调用时间的监控中心 |
| Container | 服务运行容器 |

调用关系说明

服务容器负责启动，加载，运行服务提供者。

服务提供者在启动时，向注册中心注册自己提供的服务。

服务消费者在启动时，向注册中心订阅自己所需的服务。

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。

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。

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。

1. 用法

服务提供者:

* 1. 定义服务接口, 单独打包, 在消费方和接收方共享.
  2. 在服务提供方法实现接口
  3. 在服务提供方配置声明暴露服务

<dubbo:application name=””/> 提供方应用信息

<dubbo:registry address=””/> 注册中心地址

<dubbo:protocol name=”” port=”20880”/> 服务暴露端口

<dubbo:service interface=”” ref=””/> 服务接口及实现bean的引用

<bean id=”” class=””/> 服务接口实现bean配置

服务消费者:

* 1. 服务消费者Spring配置

<dubbo:application name=””/> 消费方应用信息

<dubbo:registry address=””/> 注册中心地址

<dubbo:reference id=”” interface=””/> 服务接口, 类似于本地bean配置.

1. 推荐用法
   1. 在provider上尽量多配置consumer属性.一是因为提供者比消费者更清楚服务性能,二是可以为consumer提供缺省值.因为consumer配置是全局性的,减少不确定性.

provider上配置的属性有:

timeout: 方法调用超时

retries: 失败重试次数, 默认为2

loadbalance: 负载均衡算法,默认为random,轮询roundrobin,最不活跃优先 leastactive.

actives: 消费端最大并发调用限制.达到上限后新调用会等待直到超时(针对方法和服务)

* 1. provider上合理配置provider端属性

treads 服务线程池大小

executes 一个服务提供者并行请求上限.

* 1. 配置管理信息 owner配置负责人
  2. 配置Dubbo缓存文件

提供者列表缓存<dubbo:registry file=”${user.home}/output/dubbo.cache” />

注意: 文件路径,保证文件不会在发布过程中被删除, 多个应用进程不使用同一个文件,以免被覆盖.

这个配置是当应用重启中,Dubbo注册中心不可用时应用会从这个缓存文件读取服务提供者列表的信息, 以保证应用的可靠性.

* 1. 监控配置

使用固定端口暴露服务, 保证注册中心有延迟的情况下, 消费者通过缓存列表也能调用到原地址, 保证调用成功.

* 1. 推荐使用xml配置. 不使用properties文件.

1. 配置的覆盖规则: 方法级别优于接口级别, consumer 优于 provider ,优于默认配置.