分布式服务框架dubbo

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

LOGO

目录

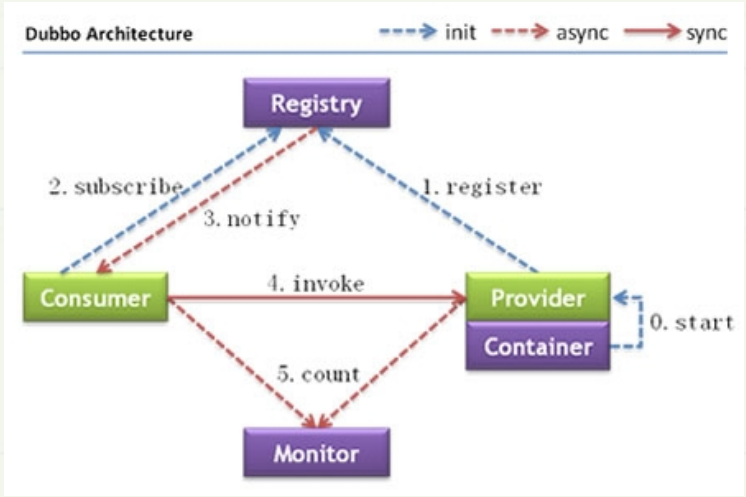
- dubbo简介
- dubbo架构
- 配置方式
- 注册中心
- 协议
- NIO
- Serialization
- 容错
- LoadBalance
- 例子

dubbo简介

- 是什么
  - 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
- 做什么
  - 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。
  - 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。
  - 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架构

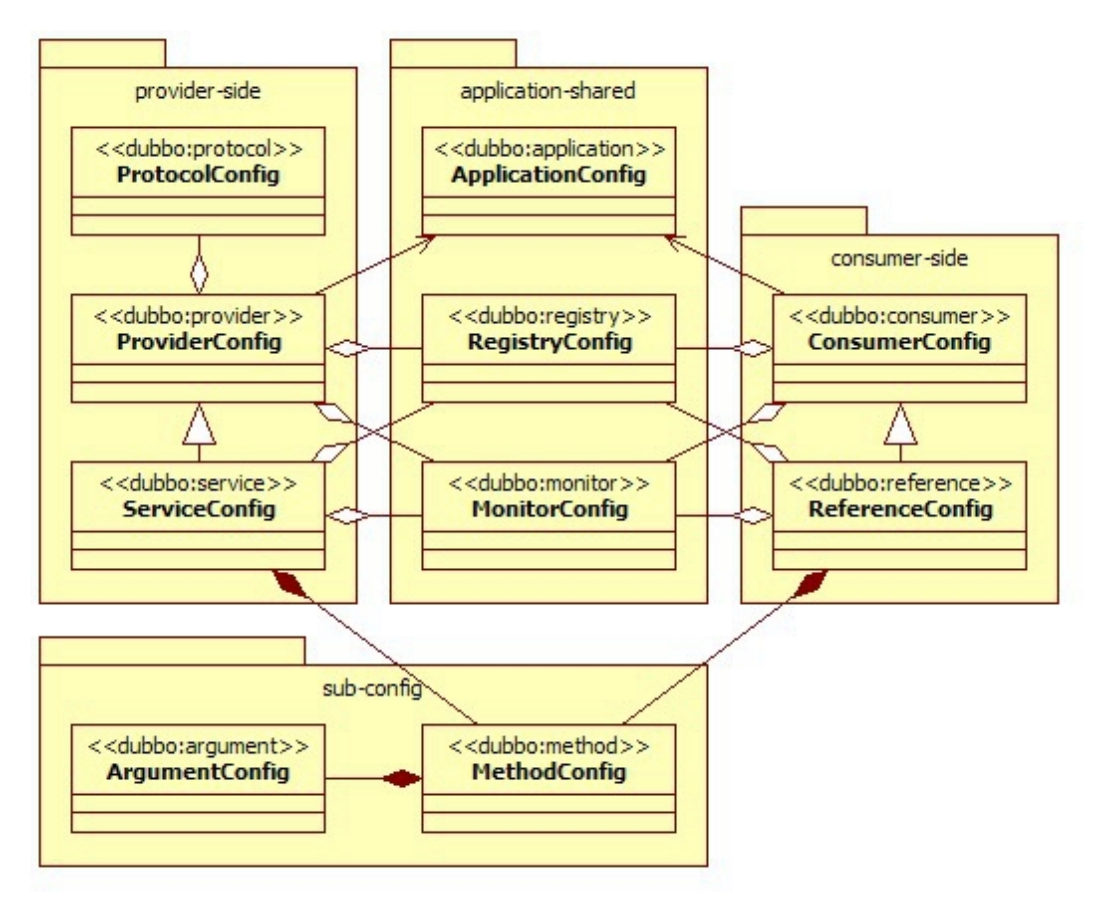
- 几个概念
  - api
  - provider
  - consumer
  - registry
  - container
  - monitor
- 流程
  - container启动，加载provider
  - provider注册服务到registry
  - consumer从registry订阅服务
  - registry通知consumer：provider地址列表
  - consumer远程调用provider提供的服务：软负载
  - provider返回结果
  - consumer和provider定时发送统计数据到monitor



配置方式

- jvm
- xml
- property
- annotation
- API

- jvm>xml>property
- consumer优先



配置方式-XML

- <dubbo:application/> 配置当前应用信息
- <dubbo:monitor/> 用于配置连接监控中心相关信息，可选
- <dubbo:protocol/> 协议配置由提供方指定，消费方被动接受
- <dubbo:registry/> 用于配置连接注册中心相关信息
- <dubbo:provider/> 提供方的缺省值，当ProtocolConfig和ServiceConfig某属性没有配置时，采用此缺省值，可选
- <dubbo:consumer/> 消费方缺省配置，当ReferenceConfig某属性没有配置时，采用此缺省值，可选
- <dubbo:module/> 模块配置，用于配置当前模块信息，可选
- <dubbo:service/> 用于暴露服务，一个服务可以用多个协议暴露，也可以注册到多个注册中心。
- <dubbo:reference/> 用于创建一个远程服务代理，可以指向多个注册中心。
- <dubbo:method/> 用于ServiceConfig和ReferenceConfig指定方法级的配置信息。
- <dubbo:argument/> 用于指定方法参数配置。

配置方式-property

- 如果公共配置很简单，没有多注册中心，多协议等情况，或者想多个Spring容器想共享配置，可以使用dubbo.properties作为缺省配置
- Dubbo将自动加载classpath根目录下的dubbo.properties，可以通过JVM启动参数：-Ddubbo.properties.file=xxx.properties 改变缺省配置位置

- dubbo.application.name=foo
- dubbo.application.owner=bar
- dubbo.registry.address=10.20.153.10:9090

配置方式-annotation

- <dubbo:annotation package="com.xx.xxx.xxxx" />
- @Service(version="1.0.0")
- @Reference(version="1.0.0")

配置方式-API

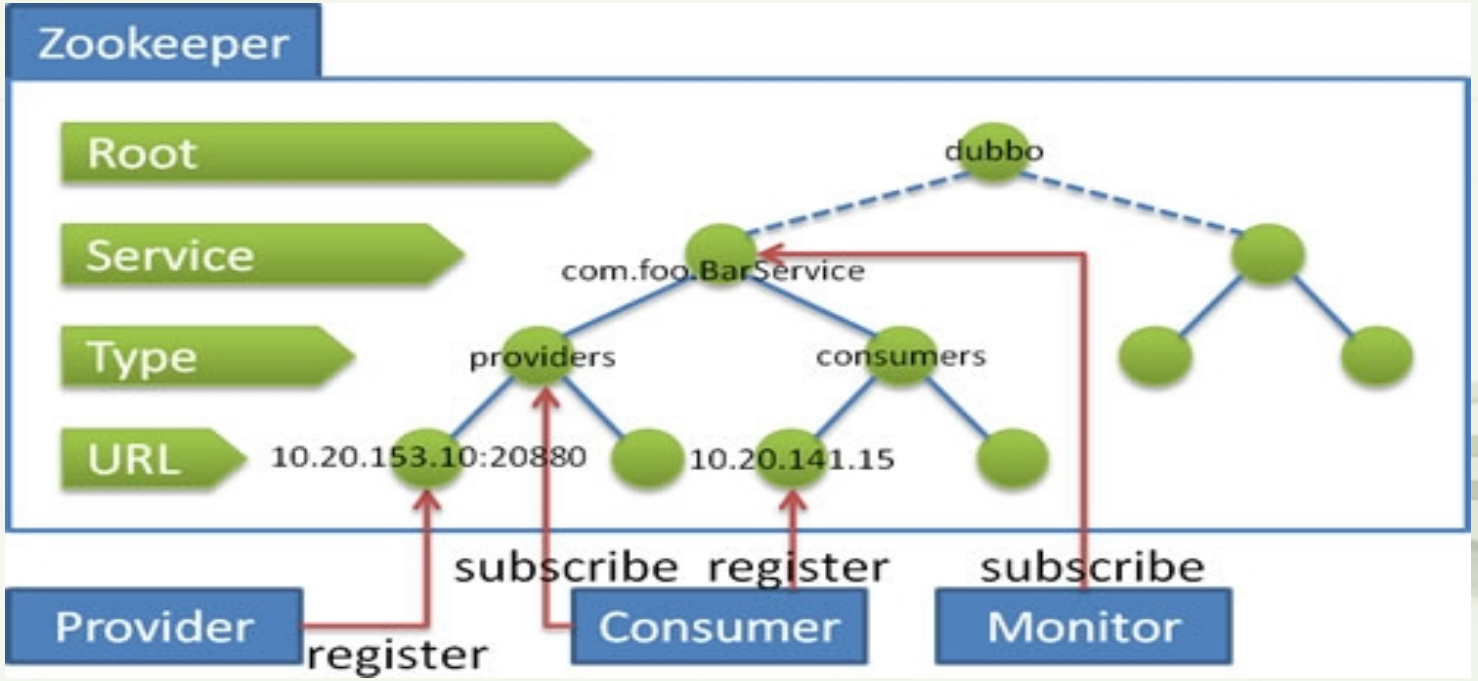
- ApplicationConfig
- RegistryConfig
- ProtocolConfig
- ServiceConfig
- ReferenceConfig

注册中心

- Zookeeper
- Redis
- Multicast
- Simple

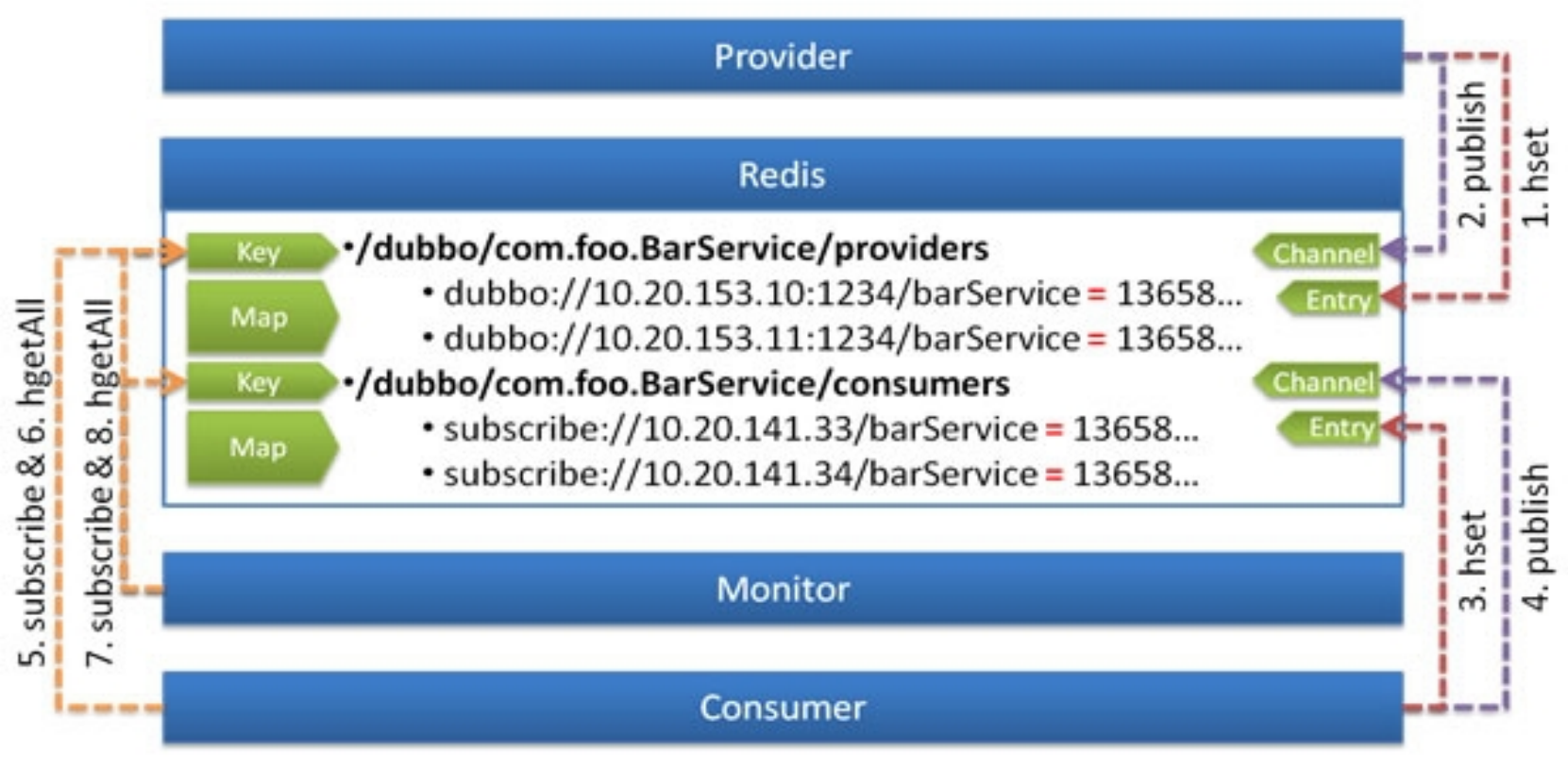
注册中心-Zookeeper

- Apacahe Hadoop的子项目，是一个树型的目录服务
- 客户端
  - zkclient（默认）
  - Curator（client="curator"）
- provider启动时向/dubbo/com.foo.BarService/providers目录下写入自己的URL地址。
- consumer者启动时订阅/dubbo/com.foo.BarService/providers目录下的提供者provider URL地址。并向/dubbo/com.foo.BarService/consumers目录下写入自己的URL地址



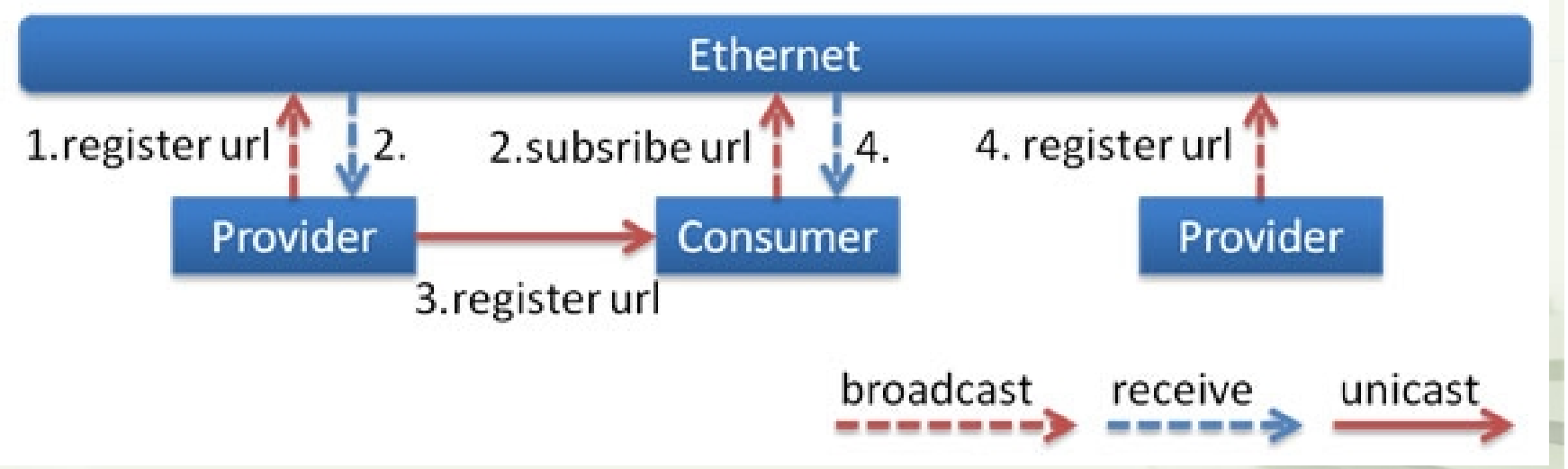
注册中心-Redis

- Key/Map
- 主Key为服务名和类型。
- Map中的Key为URL地址。
- Map中的Value为过期时间，用于判断脏数据，脏数据由监控中心删除。
  (注意：服务器时间必需同步，否则过期检测会不准确)



注册中心-Multicast

- 不需要启动任何中心节点，只要广播地址一样，就可以互相发现
  - 224.0.0.0 - 239.255.255.255
- 组播受网络结构限制，只适合小规模应用或开发阶段使用
- 默认provider单播给consumer，如果一个机器上同时启了多个consumer进程，consumer需声明unicast=false，否则只会有一个consumer能收到消息



注册中心-Simple

- 注册中心本身就是一个普通的Dubbo服务，不支持集群，不适合直接用于生产环境

```
<dubbo:application name="simple-registry" />
  <!-- 暴露服务协议配置 -->
  <dubbo:protocol port="9090" />
  <!-- 暴露服务配置 -->
  <dubbo:service interface="com.alibaba.dubbo.registry.RegistryService"
   ref="registryService" registry="N/A" ondisconnect="disconnect"
   callbacks="1000">
      <dubbo:method name="subscribe"><dubbo:argument index="1"
   callback="true" /></dubbo:method>
      <dubbo:method name="unsubscribe"><dubbo:argument index="1"
   callback="false" /></dubbo:method>
  </dubbo:service>
  <!-- 简单注册中心实现，可自行扩展实现集群和状态同步 -->
  <bean id="registryService"
   class="com.alibaba.dubbo.registry.simple.SimpleRegistryService" />
```

协议

- Dubbo（默认）
- Rmi
- Hessian
- Http
- Webservice
- Thrift
- Memcached
- Redis

协议-Dubbo

- 连接个数：单连接
- 连接方式：长连接
- 传输协议：TCP
- 传输方式：NIO异步传输
- 序列化：Hessian二进制序列化
- 适用范围：传入传出参数数据包较小（建议小于100K），消费者比提供者个数多，单一消费者无法压满提供者，尽量不要用dubbo协议传输大文件或超大字符串。
- 适用场景：常规远程服务方法调用

协议-Rmi

- 连接个数：多连接
- 连接方式：短连接
- 传输协议：TCP
- 传输方式：同步传输
- 序列化：Java标准二进制序列化
- 适用范围：传入传出参数数据包大小混合，消费者与提供者个数差不多，可传文件。
- 适用场景：常规远程服务方法调用，与原生RMI服务互操作

协议-Hessian

- 连接个数：多连接
- 连接方式：短连接
- 传输协议：HTTP
- 传输方式：同步传输
- 序列化：Hessian二进制序列化
- 适用范围：传入传出参数数据包较大，提供者比消费者个数多，提供者压力较大，可传文件。
- 适用场景：页面传输，文件传输，或与原生hessian服务互操作

协议-Http

- 连接个数：多连接
- 连接方式：短连接
- 传输协议：HTTP
- 传输方式：同步传输
- 序列化：表单序列化
- 适用范围：传入传出参数数据包大小混合，提供者比消费者个数多，可用浏览器查看，可用表单或URL传入参数，暂不支持传文件。
- 适用场景：需同时给应用程序和浏览器JS使用的服务。

协议-Webservice

- 连接个数：多连接
- 连接方式：短连接
- 传输协议：HTTP
- 传输方式：同步传输
- 序列化：SOAP文本序列化
- 适用场景：系统集成，跨语言调用。

协议-Thrift

- facebook -> apache
- 当前 dubbo 支持的 thrift 协议是对 thrift 原生协议的扩展，在原生协议的基础上添加了一些额外的头信息，比如service name，magic number等。使用dubbo thrift协议同样需要使用thrift的idl compiler编译生成相应的java代码
- Thrift不支持数据类型:null

协议-Memcached

- Memcached是一个高效的KV缓存服务器
- RegistryFactory registryFactory = ExtensionLoader.getExtensionLoader(RegistryFactory.class).getAdaptiveExtension();
- Registry registry = registryFactory.getRegistry(URL.valueOf("zookeeper://10.20.153.10:2181"));
- registry.register(URL.valueOf("memcached://10.20.153.11/com.foo.BarService?category=providers&dynamic=false&application=foo&group=member&loadbalance=consistenthash"));

协议-Redis

- Redis是一个高效的KV存储服务器
- RegistryFactory registryFactory = ExtensionLoader.getExtensionLoader(RegistryFactory.class).getAdaptiveExtension();
- Registry registry = registryFactory.getRegistry(URL.valueOf("zookeeper://10.20.153.10:2181"));
- registry.register(URL.valueOf("redis://10.20.153.11/com.foo.BarService?category=providers&dynamic=false&application=foo&group=member&loadbalance=consistenthash"));

NIO

- Netty
  - jboss
- Mina
  - apache
- Grizzly
  - sun

Serialization

- Hessian
- Dubbo
- Json
- Java

容错

- Failover
  - 失败自动切换，当出现失败，重试其它服务器，通常用于读操作
- Failfast
  - 快速失败，只发起一次调用，失败立即报错,通常用于非幂等性的写操作
- Failsafe
  - 失败安全，出现异常时，直接忽略，通常用于写入审计日志等操作
- Failback
  - 失败自动恢复，后台记录失败请求，定时重发，通常用于消息通知操作
- Forking
  - 并行调用多个服务器，只要一个成功即返回，通常用于实时性要求较高的读操作
- Broadcast
  - 广播调用所有提供者，逐个调用，任意一台报错则报错，通常用于更新提供方本地状态

LoadBalance

- Random
  - 随机，按权重设置随机概率
- RoundRobin
  - 轮循，按公约后的权重设置轮循比率
- 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，使慢的机器收到更少请求
- ConsistentHash
  - 一致性Hash，相同参数的请求总是发到同一提供者，当某一台提供者挂时，原本发往该提供者的请求，基于虚拟节点，平摊到其它提供者，不会引起剧烈变动

谢谢观赏

@苦逼少侠