Dubbo培训

技术部. 甘建新.2018.03.24

1. Dubbo是什么
2. Dubbo的原理
3. Zookeeper注册中心
4. 业务系统改造
5. 错误码

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。

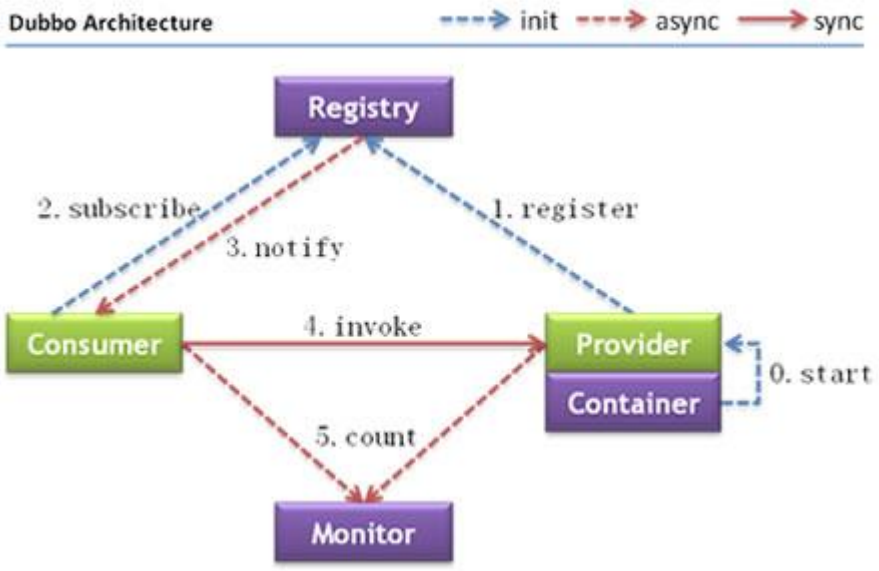
Dubbo核心部分

■ 远程通讯：提供对多种框架抽象封装，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的原理



节点	角色说明
Provider	暴露服务的提供方
Consumer	调用远程服务的服务消费方
Registry	服务注册和发现的注册中心
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

网络权限

网段	描述
10.228.4.X	对内WEB
10.228.5.X	对外WEB
10.228.6.X	应用区
10.228.7.X	数据库区
10.228.10.X	风控应用区

注：跨网段接口需要开通，目标端口包括：provider端口，zookeeper端口，monitor端口

集群容错

- **Failover Cluster（默认）**

失败自动切换，当出现失败，重试其它服务器 1。通常用于读操作，但重试会带来更长延迟。可通过 retries="2" 来设置重试次数(不含第一次)。

- **Failfast Cluster**

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，比如新增记录。

- **Failsafe Cluster**

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 Cluster**

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 Cluster**

并行调用多个服务器，只要一个成功即返回。

- **Broadcast Cluster**

广播调用所有提供者，逐个调用，任意一台报错则报错。(2.1.0开始支持)

负载策略

- Random LoadBalance

随机，按权重设置随机概率。
在一个截面上碰撞的概率高，但调用量越大分布越均匀，而且按概率使用权重后也比较均匀，有利于动态调整提供者权重。

- RoundRobin LoadBalance

轮循，按公约后的权重设置轮循比率。
存在慢的提供者累积请求问题，比如：第二台机器很慢，但没挂，当请求调到第二台时就卡在那，久而久之，所有请求都卡在调到第二台上。

- LeastActive LoadBalance

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 LoadBalance

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。

服务降级

服务端系统参数设置 | 消费端系统参数设置 | 服务降级设置 | 自定义参数

服务降级

所有方法的Mock值：

容错 ▼

容错
屏蔽

Mock返回结果

示例: return null/empty/JSON或throw com.foo.BarException

新增方法

保存

直连提供者

- **通过 XML 配置**

```
<dubbo:reference id="xxxService" interface="com.alibaba.xxx.XxxService"
url="dubbo://localhost:20890" />
```

- **通过 -D 参数指定**

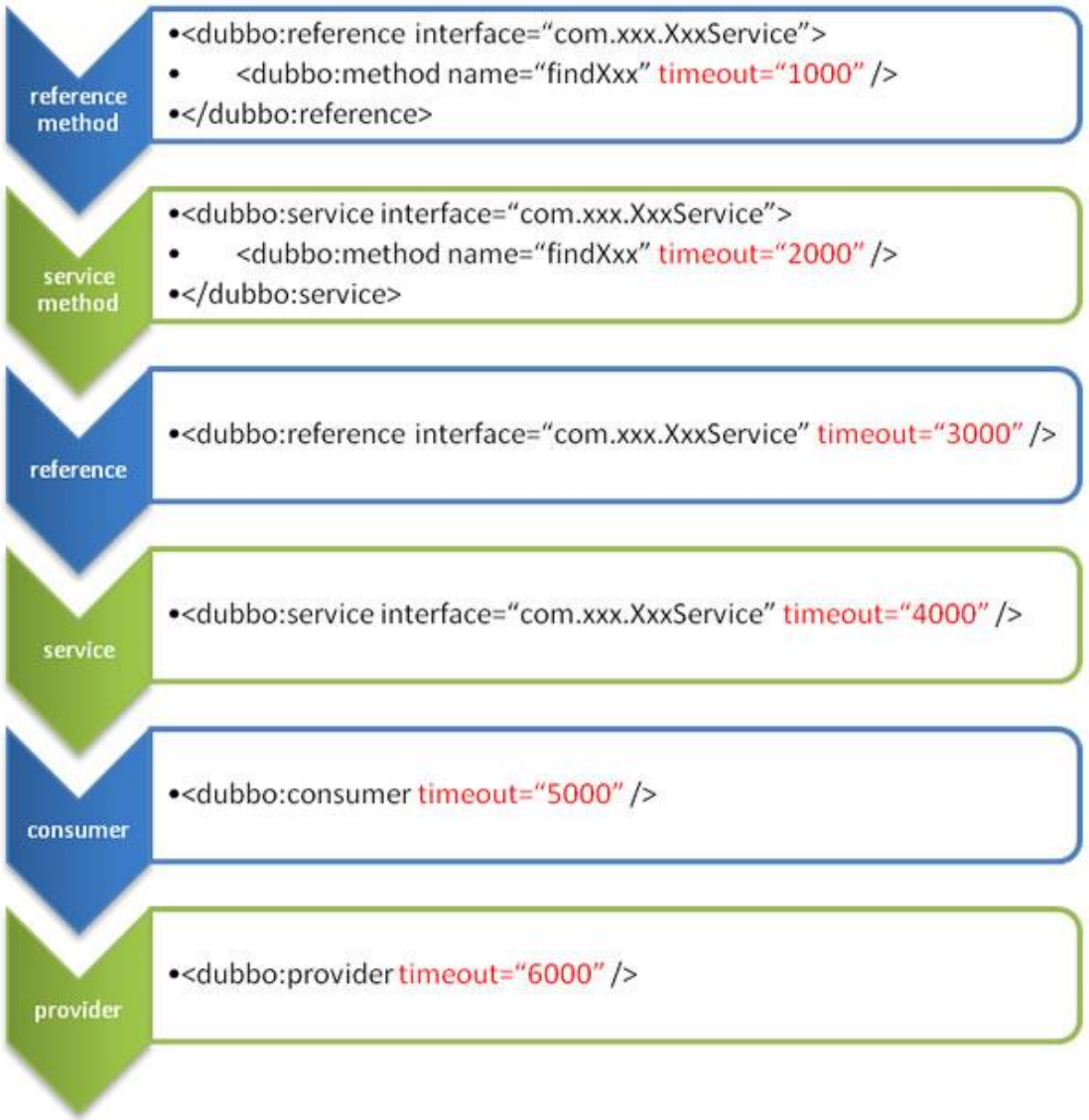
```
java -Dcom.alibaba.xxx.XxxService=dubbo://localhost:20890
```

- **通过文件映射**

```
java -Ddubbo.resolve.file=xxx.properties
com.alibaba.xxx.XxxService=dubbo://localhost:20890
```

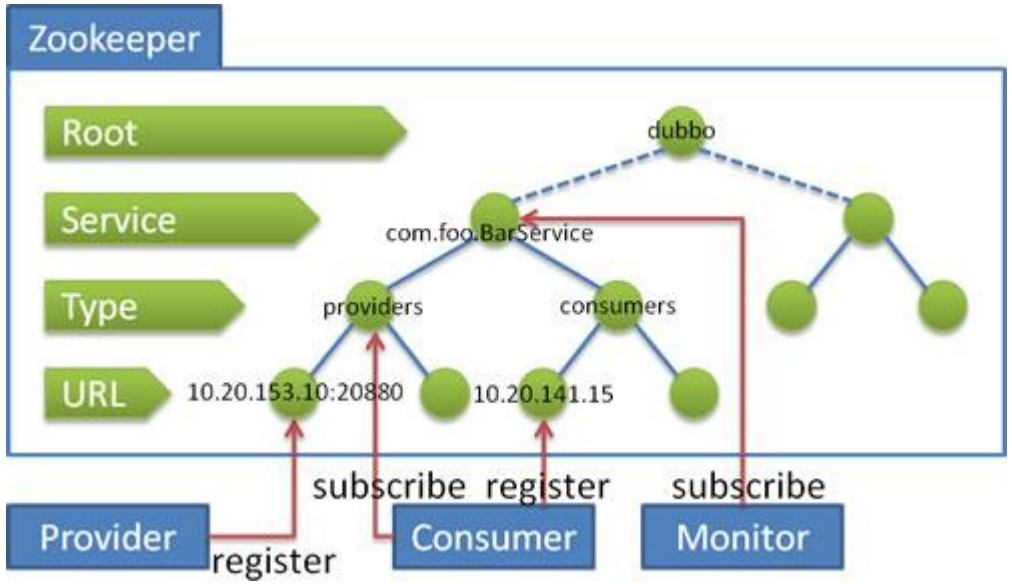
配置覆盖原则

- 方法级优先，接口级次之，全局配置再次之。
- 如果级别一样，则消费方优先，提供方次之。



Zookeeper注册中心

Zookeeper是Apacahe Hadoop的子项目，是一个树型的目录服务，支持变更推送，适合作为Dubbo服务的注册中心，工业强度较高，可用于生产环境，并推荐使用，参见：http://zookeeper.apache.org



代码改造 pom.xml

```
<dubbo.version>2.8.4</dubbo.version>
<zkclient.version>0.1</zkclient.version>


 <!-- https://mvnrepository.com/artifact/com.alibaba/dubbo -->
<dependency>
<groupId>com.alibaba</groupId>
<artifactId>dubbo</artifactId>
<version>${dubbo.version}</version>
<exclusions>
<exclusion>
<artifactId>spring</artifactId>
<groupId>org.springframework</groupId>
</exclusion>
</exclusions>
</dependency>
<!-- zk -->
<dependency>
<groupId>com.github.sgroschupf</groupId>
<artifactId>zkclient</artifactId>
<version>${zkclient.version}</version>
</dependency>
```

代码改造 dubbo.properties

```
########################################################### 公共配置
## 应用名称
dubbo.application.name=payment
## 应用所属者
dubbo.application.owner=payment
## 应用机构名称
dubbo.application.organization=kjtpay
## 应用版本
dubbo.application.version=1.0.0
## 注册中心地址
dubbo.registry.address=zookeeper://192.168.180.42:2181?backup=192.168.180.43:2181,192.168.180.44:2181
## 缓存文件地址
dubbo.registry.file=/home/localadmin/.dubbo/dpm-manager.cache
########################################################### 生产者配置
## 生产者协议名称
dubbo.provider.protocol.name=dubbo
## 生产者协议端口，该端口规则为 2+当前tomcat端口，示例26003
dubbo.provider.protocol.port=26003
## 生产者线程数量
dubbo.provider.protocol.threads=100
## 生产者配置超时时间，消费者若配置超时时间会覆盖该值
dubbo.provider.timeout=60000
## 重试次数
dubbo.provider.retries=0
########################################################### 消费者配置
## 超时时间
dubbo.consumer.timeout=15000
## 重试次数
```

代码改造 application-dubbo.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://code.alibabatech.com/schema/dubbo
http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

        <!-- 提供方应用信息，用于计算依赖关系 -->
        <dubbo:application name="${dubbo.application.name}"
version="${dubbo.application.version}"
owner="${dubbo.application.owner}" organization="${dubbo.application.organization}" />

        <!-- 使用zookeeper注册中心暴露服务地址 -->
        <dubbo:registry address="${dubbo.registry.address}"  file="${dubbo.registry.file}"/>

        <!-- 监控 -->
        <dubbo:monitor protocol="registry" />

</beans>
```

代码改造 dubbo-provider.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://code.alibabatech.com/schema/dubbo
http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

        <!-- 生产者公用设置 -->
        <dubbo:provider timeout="${dubbo.provider.timeout}"
retries="${dubbo.provider.retries}"></dubbo:provider>
        <!-- 用dubbo协议在20880端口暴露服务 -->
        <dubbo:protocol name="${dubbo.provider.protocol.name}"
port="${dubbo.provider.protocol.port}"
                threads="${dubbo.provider.protocol.threads}" />

        <!-- 声明需要暴露的服务接口 -->
        <dubbo:service interface="com.netfinworks.payment.service.facade.BasicConfigQueryFacade"
                ref="basicConfigQueryFacade" />
</beans>
```

代码改造 dubbo-consumer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://code.alibabatech.com/schema/dubbo
http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

        <!-- 设置超时 -->
        <dubbo:consumer timeout="${dubbo.consumer.timeout}" retries="${dubbo.consumer.retries}"
loadbalance="roundrobin"/>

        <!-- 调用服务 -->
        <dubbo:reference id="fundRequestFacade"
interface="com.netfinworks.cmf.fss.service.facade.api.FundRequestFacade" />
        <dubbo:reference id="orderQueryFacade"
interface="com.netfinworks.cmf.fss.service.facade.api.OrderQueryFacade" />
        <dubbo:reference id="controlRequestFacade"
interface="com.netfinworks.cmf.fss.service.facade.api.ControlRequestFacade" />

</beans>
```

代码改造 application.xml

```
<import resource="classpath:META-INF/spring/applicationContext-dubbo.xml" />
<import resource="classpath:META-INF/spring/dubbo-provider.xml" />
<import resource="classpath:META-INF/spring/dubbo-consumer.xml" />
```

系统错误码使用规范

系统错误码使用规范：

- 构建新系统先现在cmdb中的系统信息中定义好错误码前缀。并经主管、架构审核；
- 错误码前缀规范：[A-Z]{2,6}_$。示例：CMF_　　TSS_；
- 服务接口返回包含2层错误信息，服务本身的错误信息(resultCode,resultMessage)；原始错误信息（unityResultCode，unityResultMessage）；

  成功结果码标准：S0001

  错误码传递规范：无原始错误信息时，将服务自身错误设置到原始错误信息。调用的服务返回原始错误信息时，不能在改变，要逐层返回透传。

  示例：

  mag {"code":"F0021","msg":"处理失败","subCode":"CAP_F0031","subMsg":"验证码错误"}

产品中心错误吗映射规则进行转换：

- tss {"resultCode":"TSS_F0011","resultMessage":"处理失败","unityResultCode":"CAP_F0031","unityResultMessage":"验证码错误"}
- payment {"resultCode":"PE_F0002","resultMessage":"处理失败","unityResultCode":"CAP_F0031","unityResultMessage":"验证码错误"}
- cmf {"resultCode":"CMF_F0001","resultMessage":"处理失败","unityResultCode":"CAP_F0031","unityResultMessage":"验证码错误"}　**对渠道返回的错误吗通过统一的配置进行转换**
- channel {"apiResultCode":"01","apiResultMessage":"验证码错误","apiSubResultCode":"","apiSubResultMessage":""}

系统错误码maven坐标

```
<dependency>
  <groupId>com.kjtpay</groupId>
  <artifactId>commons-model</artifactId>
  <version>0.0.2</version>
</dependency>
```

参考资料

http://dubbo.apache.org/

http://dubbo.io/books/dubbo-user-book

http://dubbo.io/books/dubbo-dev-book/

http://dubbo.io/books/dubbo-admin-book/

https://github.com/apache/incubator-dubbo

http://zookeeper.apache.org/doc/trunk/index.html

Thank you !

海尔金控
Haier Financial Holdings