

Dubbo功能介绍

阿里巴巴-B2B-平台技术部-应用框架
2011-12 Dubbo
weibo.com/dubbo

大纲

- Dubbo简要介绍
- Dubbo-RPC基本功能
- Dubbo-RPC高级功能
- 最佳实践

Dubbo是什么

- 分布式服务框架
 - 高性能和透明化的[RPC](#)远程服务调用方案
 - [SOA](#)服务治理方案

如何使用Dubbo

- 本地服务

```
<bean id="xxxService" class="com.xxx.XxxServiceImpl" />  
<bean id="xxxAction" class="com.xxx.XxxAction">  
  <property name="xxxService" ref="xxxService" />  
</bean>
```

- 远程服务

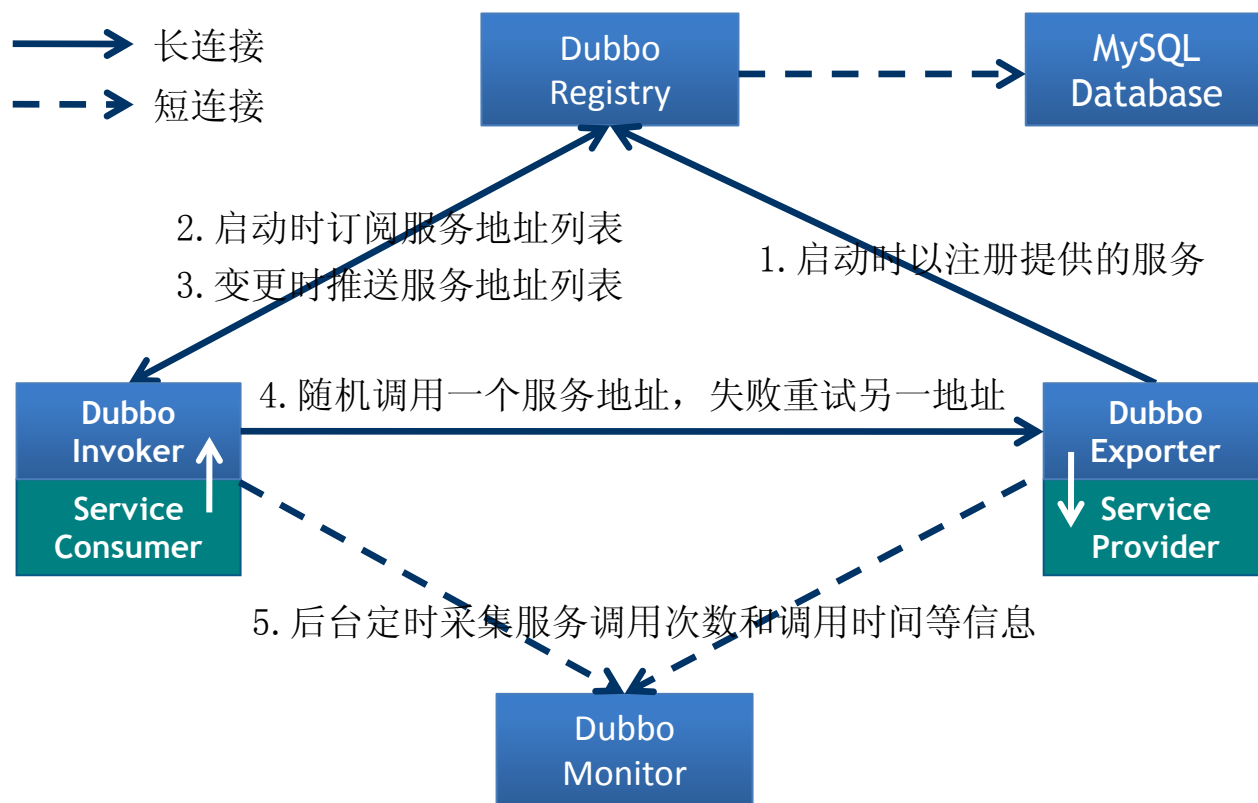
```
<bean id="xxxService" class="com.xxx.XxxServiceImpl" />  
<dubbo:service interface="com.xxx.XxxService" ref="xxxService" />
```

```
<dubbo:reference id="xxxService" interface="com.xxx.XxxService" />  
<bean id="xxxAction" class="com.xxx.XxxAction">  
  <property name="xxxService" ref="xxxService" />  
</bean>
```

Dubbo能做什么

- 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
- 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

Dubbo基本原理



Dubbo RPC 基本功能篇

SOA因你而简单

Dubbo-RPC基本功能

配置

- 配置继承
- 可配置可编程

服务匹配

- 服务分组、多版本
- 指定调用
- 只订阅

集群& 容错

- 集群原理
- 容错规则

多协议

- 不同服务不同协议
- 同一服务多协议
- 本地服务调用

多注册中心

Graceful shutdown

基本功能 - 配置继承

实际使用中发现，服务提供者比消费者更清楚一个方法的执行时间，是否允许重试等信息，所以增加允许服务提供者为消费者设置缺省值，并采用继承风格：



基本功能 - 可编程配置 - 暴露服务

解决框架集成时过度依赖于Dubbo的内部API问题，增加与配置一致的映射API：

编程配置：

```
// 服务实现
XxxService xxxService = new XxxServiceImpl();

// 当前应用配置
ApplicationConfig application = new ApplicationConfig();
application.setName("xxx");

// 连接注册中心配置
RegistryConfig registry = new RegistryConfig();
registry.setAddress("10.20.130.230:9090");
registry.setUsername("aaa");
registry.setPassword("bbb");

// 服务提供者协议配置
ProviderConfig provider = new ProviderConfig();
provider.setProtocol("dubbo");
provider.setPort(12345);
provider.setThreads(200);

// 服务提供者暴露服务配置
ServiceConfig service = new ServiceConfig();
service.setApplication(application);
service.setRegistry(registry); // 多个注册中心可以用setRegistries()
service.setProvider(provider); // 多个提供者可以用setProviders()
service.setInterfaceClass(XxxService.class);
service.setRef(xxxService);
service.setVersion("1.0.0");
service.export(); // 触发服务注册
```

Schema配置：

```
<?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://repo.alibaba-inc.com/schema/dubbo"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://repo.alibaba-inc.com/schema/dubbo
http://repo.alibaba-inc.com/schema/dubbo/dubbo-component-2.0.xsd">
  <!--服务实现 -->
  <bean id="xxxService" class="com.alibaba.xxx.XxxServiceImpl" />

  <!--当前应用配置 -->
  <dubbo:application name="morgan" />

  <!-- 连接注册中心配置 -->
  <dubbo:registry address="10.20.130.230:9090" username="admin"
password="hello1234" />

  <!-- 服务提供者协议配置 -->
  <dubbo:provider protocol="dubbo" port="12345" threads="200" />

  <!-- 服务提供者暴露服务配置 -->
  <dubbo:service interface="com.alibaba.xxx.XxxService"
version="1.0.0" ref="xxxService" />
</beans>
```

基本功能 - 可编程配置- 引用服务

编程配置:

```
// 当前应用配置
ApplicationConfig application = new ApplicationConfig();
application.setName("yyy");

// 连接注册中心配置
RegistryConfig registry = new RegistryConfig();
registry.setAddress("10.20.130.230:9090");
registry.setUsername("aaa");
registry.setPassword("bbb");

// 服务消费者缺省值配置
ConsumerConfig consumer = new ConsumerConfig();
consumer.setTimeout(5000);
consumer.setRetries(2);

// 引用远程服务
ReferenceConfig reference = new ReferenceConfig();
reference.setApplication(application);
reference.setRegistry(registry); // 多个注册中心可以用setRegistries()
reference.setConsumer(consumer);
reference.setInterfaceClass(XxxService.class);
reference.setVersion("1.0.0");
XxxService xxxService = reference.get(); // 获取远程xxxService代理
```

Schema配置:

```
<?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://repo.alibaba-inc.com/schema/dubbo"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://repo.alibaba-inc.com/schema/dubbo
http://repo.alibaba-inc.com/schema/dubbo/dubbo-component-2.0.xsd">
    <!-- 当前应用信息配置 -->
    <dubbo:application name="kylin" />

    <!-- 连接注册中心配置 -->
    <dubbo:registry address="10.20.130.230:9090" username="admin"
password="hello1234" />

    <!-- 服务消费者缺省值配置 -->
    <dubbo:consumer timeout="5000" retries="2" />

    <!-- 引用远程服务 -->
    <dubbo:reference id="xxxService"
interface="com.alibaba.xxx.XxxService" version="1.0.0" />
</beans>
```

基本功能- 服务分组

当一个接口有多种实现时，可以用group区分

服务提供者

```
<dubbo:service group="feedback" interface="com.xxx.IndexService" />  
<dubbo:service group="member" interface="com.xxx.IndexService" />
```

服务消费者

```
<dubbo:reference id="fservice" group="feedback" interface="com.xxx.IndexService" />  
<dubbo:reference id="mservice" group="member" interface="com.xxx.IndexService" />
```

基本功能- 服务分组

当一个接口实现，出现不兼容升级时，可以用版本号过渡，版本号不同的服务相互间不引用

服务提供者

```
<dubbo:service interface="com.foo.BarService" version="1.0.0" />  
<dubbo:service interface="com.foo.BarService" version="2.0.0" />
```

服务消费者

```
<dubbo:reference id="barService" interface="com.foo.BarService" version="1.0.0" />  
<dubbo:reference id="barService" interface="com.foo.BarService" version="2.0.0" />
```

基本功能- 指定调用

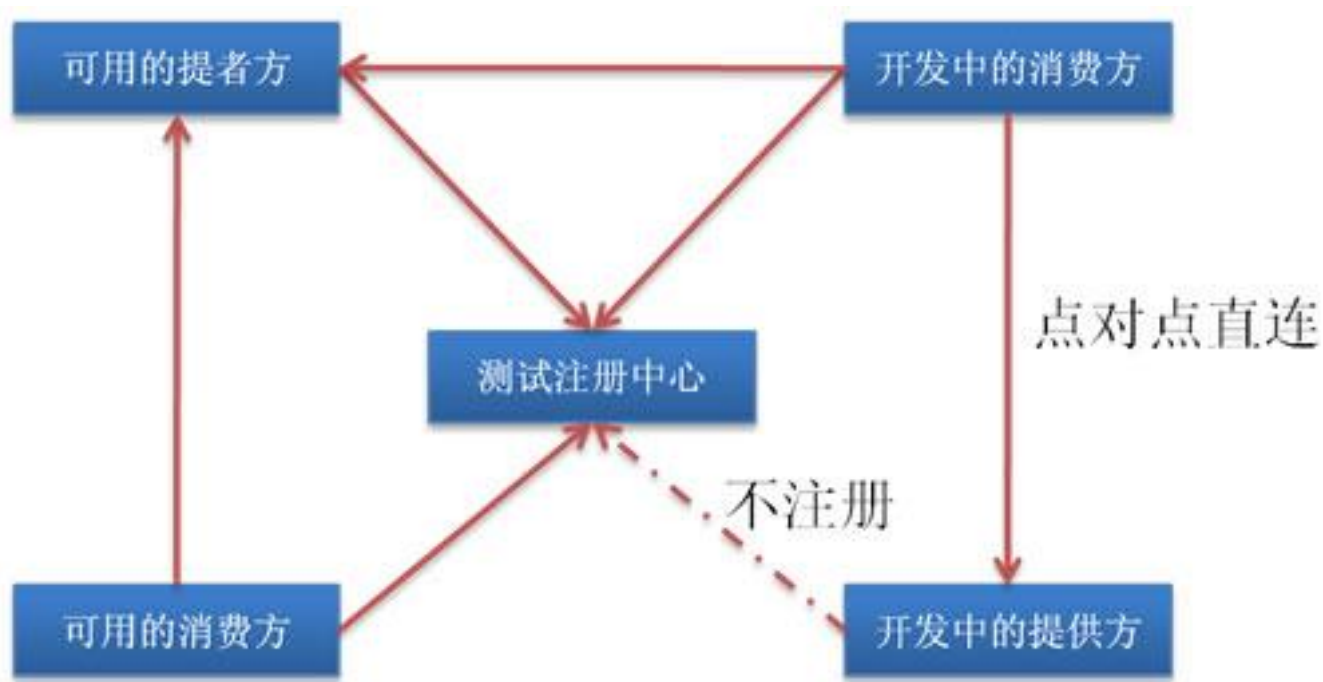
点对点直连/指定调用需求（开发/测试环境）

- 方式一 Spring配置
 - `<dubbo:reference interface="com.alibaba.xxx.XxxService" url="dubbo://localhost:20890" />`
- 方式二 Java -D参数方式
 - `java -Dcom.alibaba.xxx.XxxService=dubbo://localhost:20890`
- 方式三 映射文件方式
 - `java -Ddubbo.resolve.file=xxx.properties`
 - `com.alibaba.xxx.XxxService=dubbo://localhost:20890`
- 方式四 路由方式(future)
 - 参数匹配 host、classifier
- 方式五 修改version
 - 不推荐

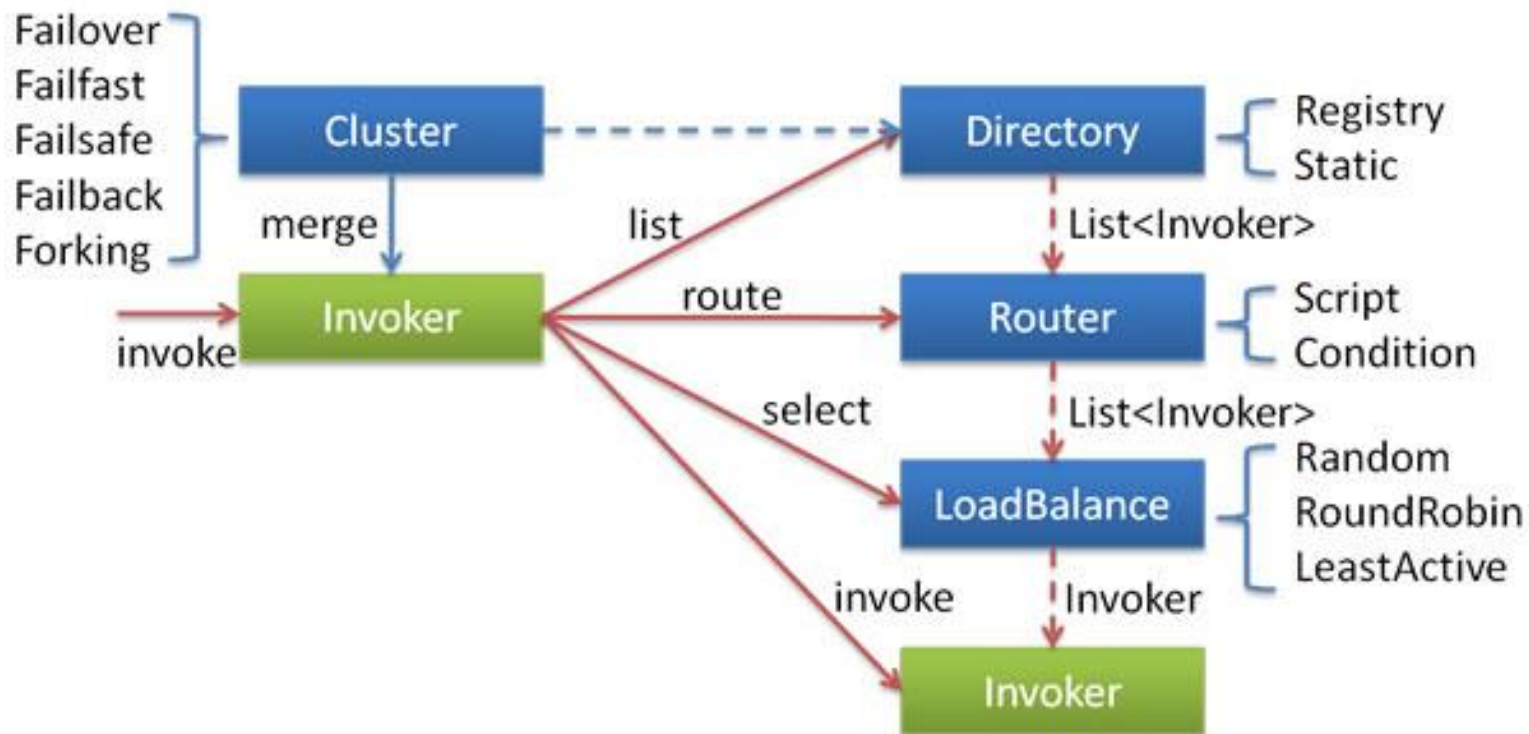
基本功能- 只订阅

共用注册中心，开发人员机器上的服务提供者被误调，影响其他开发人员（开发、测试环境）

`<dubbo:registry register="false" />`



基本功能- 集群&容错



基本功能- 多协议

- 不同服务不同协议

比如：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。

```
<!-- 多协议配置 -->
<dubbo:protocol name="dubbo" port="20880" />
<dubbo:protocol name="rmi" port="1099" />

<!-- 使用dubbo协议暴露服务 -->
<dubbo:service interface="com.alibaba.hello.api.HelloService" version="1.0.0" ref="helloService" protocol="dubbo" />
<!-- 使用rmi协议暴露服务 -->
<dubbo:service interface="com.alibaba.hello.api.DemoService" version="1.0.0" ref="demoService" protocol="rmi" />
```

- 同一服务多协议暴露

比如：需要与http客户端互操作

```
<!-- 多协议配置 -->
<dubbo:protocol name="dubbo" port="20880" />
<dubbo:protocol name="hessian" port="8080" />

<!-- 使用多个协议暴露服务 -->
<dubbo:reference id="helloService" interface="com.alibaba.hello.api.HelloService" version="1.0.0"
protocol="dubbo,hessian" />
```

基本功能 – 多协议

- InJvm调用

- 同一个jvm内部的服务调用采用短路的方式
- 先本地服务化，再做物理远程调用

```
<dubbo:protocol name="injvm" >  
<dubbo:provider interface = "HelloService" injvm="true"/>  
<dubbo:consumer interface="HelloService" injvm="true" >
```

基本功能 - 多注册中心引用

- 解决CRM需同时调用中文站和国际站PC2相同接口相同版本服务的问题

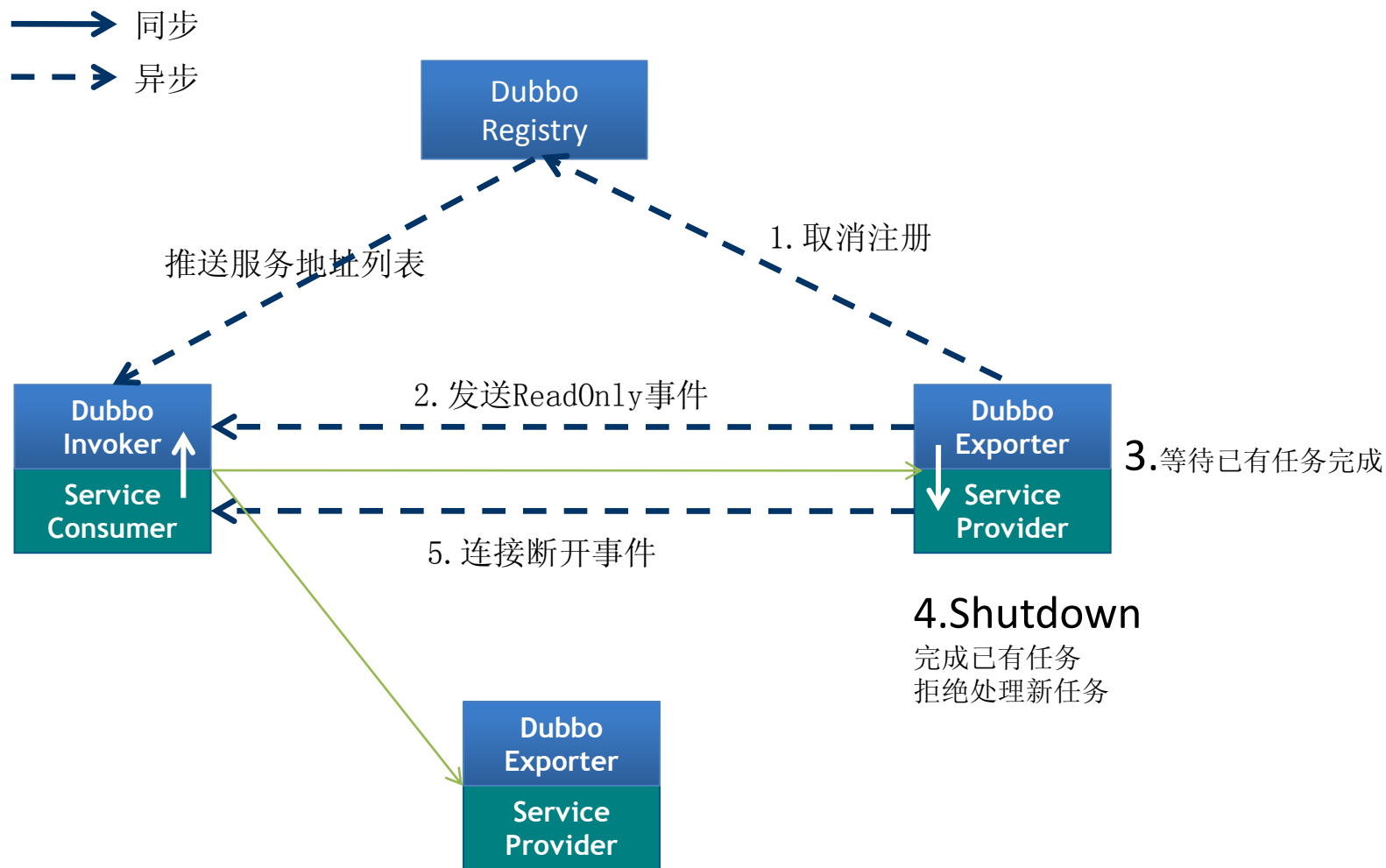
定义多个注册中心：

```
<dubbo:registry id="chinaRegistry" address="172.29.63.18:9090" />  
<dubbo:registry id="intlRegistry" address="172.29.61.132:9090" />
```

不同注册中心使用不同引用：

```
<dubbo:reference id="chinaXxxService" interface="com.alibaba.xxx.XxxService"  
version="1.0.0" registry="chinaRegistry" />  
<dubbo:reference id="intlXxxService" interface="com.alibaba.xxx.XxxService"  
version="1.0.0" registry="intlRegistry" />
```

基本功能- Graceful Shutdown



Dubbo RPC 高级功能篇

SOA因你而不同

Dubbo-RPC高级功能

Consumer Only

- 泛化调用
- Stub & Mock
- 异步调用& Forking
- 路由

Provider Only

- Telnet
- 流量控制

Consumer & Provider

- Callback
- 隐式传参

高级功能- Telnet互操作

```
C:\> telnet localhost 20880
dubbo> help
ls -l
ps -l
status -l
trace XxxService 10
count XxxService
invoke XxxService.xxxMethod(args)
log 100
```

高级功能 – 泛化调用

客户端没有业务API，弱类型泛化调用，POJO自动转为Map

配置：

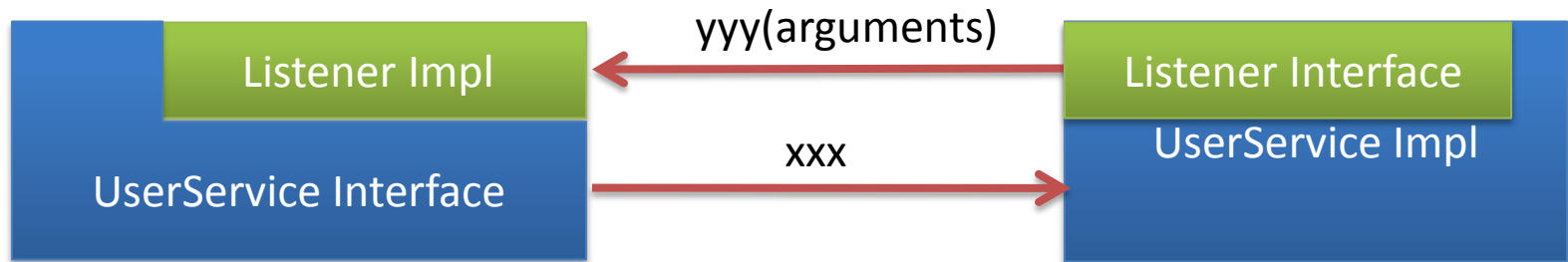
```
<dubbo:reference id="xxxService" interface="com.alibaba.xxx.XxxService"
version="1.0.0" generic="true" />
```

引用接口：(有API侵入)

```
com.alibaba.dubbo.rpc.service.GenericService
GenericService xxxService = (GenericService)context.getBean("xxxService");
```


高级功能- 显示回调

- 原理



- 使用场景

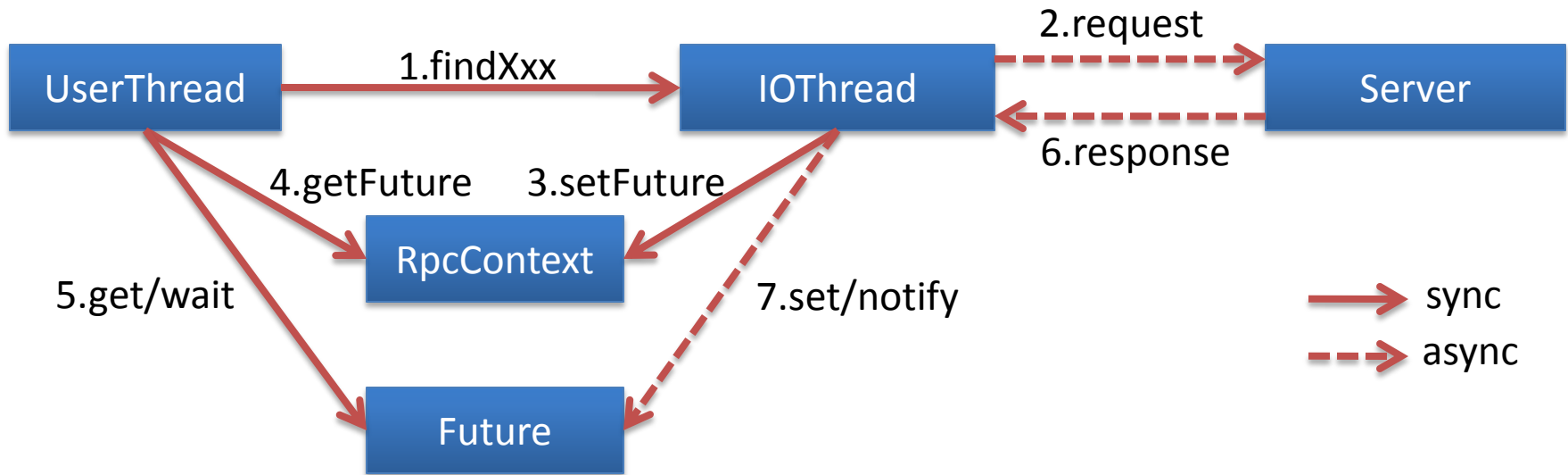
注册中心

Normandy

配合stub做热数据缓存

高级功能-异步调用

- 并行发起多个请求，但只使用一个线程
 - `<dubbo:method name=" findXxx " async="true" />`
 - `xxxService.findXxx();`
 - `Future<Xxx> xxxFuture = RpcContext.getFuture();`

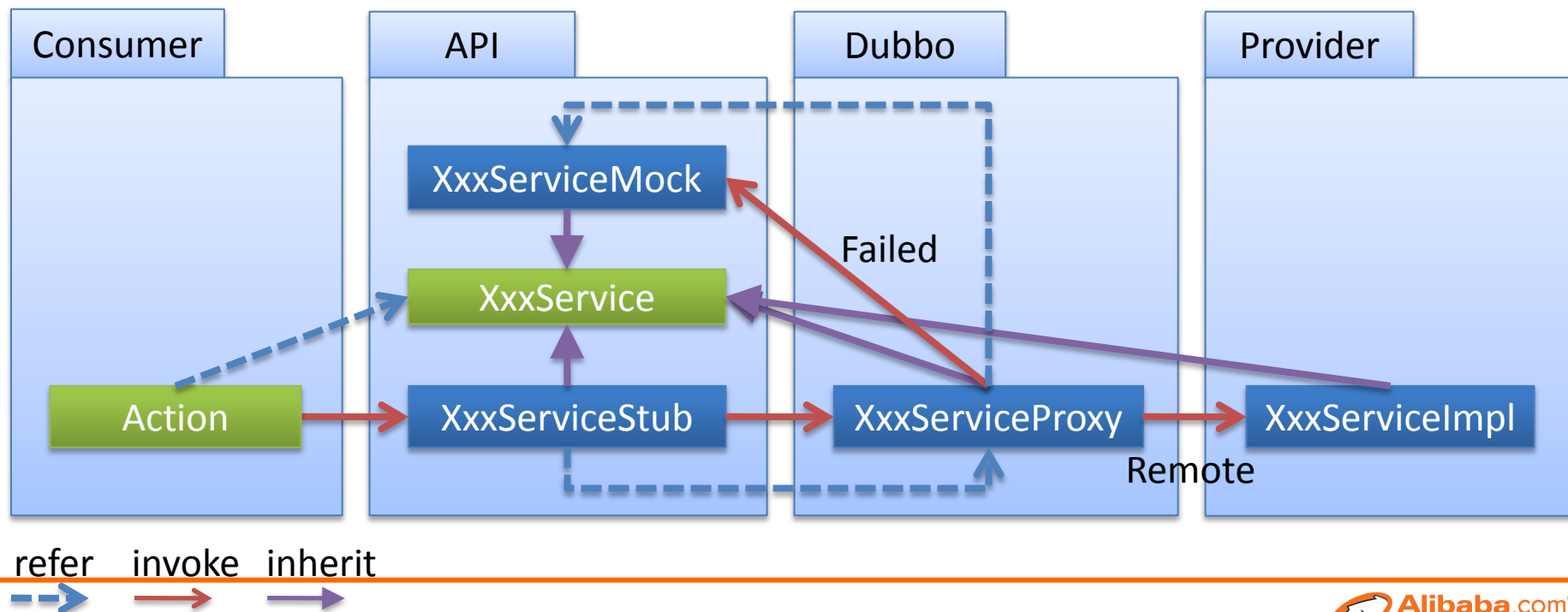


高级功能-框架事件

- <Dubbo:service ... ondisconnected= xxx >
- <Dubbo:reference ... ondisconnected =xxx>

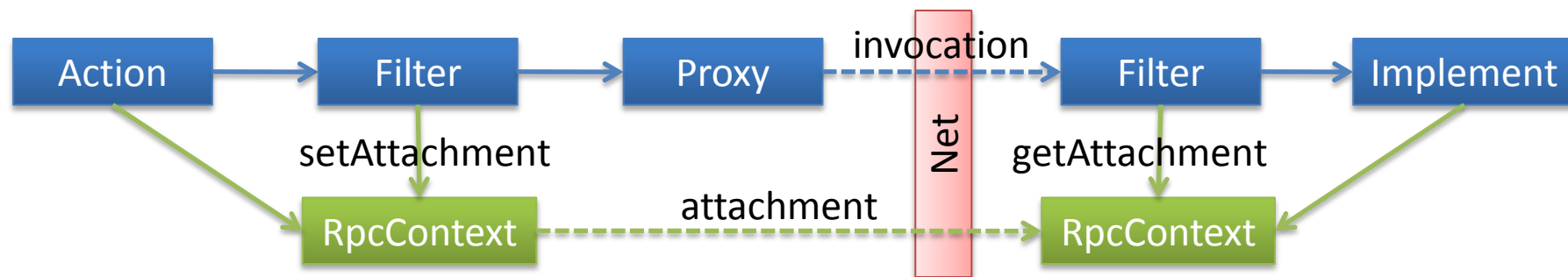
高级功能- 本地执行

- 在客户端执行部分代码，比如：
 - 在客户端缓存已查询过的数据
 - 当服务器端全部不可用时，伪造容错数据



高级功能- 隐式传参

- 隐式传参，比如：
 - 验权扩展点需要携带一些用户凭证信息
- ThreadLocal Context
 - `RpcContext.setAttachment("password", "xxx");`
 - `RpcContext.getAttachment("password");`



高级功能 – 路由

- 注册中心路由
 - 注册中心根据路由规则挑选服务提供者列表
- RPC路由
 - 类 方法 参数 级别的路由规则
 - 数据sharding
 - 开发阶段的服务过滤(classifier)
 - 可扩展的路由接口&基于ScriptEngine的实现
 - simpleEL
 - Groovy ..

高级功能 - 路由示例

JavaScript路由示例:

```
function route(invokers, invocation, context){
    var result = new java.util.ArrayList();
    if (invokers.size() > 1 &&
        invocation.getMethodName().equals("method1")) {
        result.add(invokers.get(0));
    } else {
        result.add(invokers.get(1));
    }
    return result;
};
route(invokers, invocation, context);
```

高级功能 – 流量控制

- `actives` : Consumer并发数限制
- `executes` : Provider并发数上限
- `connections` : Consumer的连接数
- `accepts` : Provider的连接上限
短连接是连接上限，长连接则是启用的连接数
- `LeastActive LoadBalance` :
调用并发数最小的Provider（从Consumer端并发）
调节Provider间并发

Dubbo-RPC - Napoli

- Use Napoli As Dubbo
 - `<dubbo:protocol name=napoli />`
 - `<dubbo:service ...protocol =napoli>`
 - `<dubbo:reference interface=XXXService>`
- Use ... As Dubbo

服务化最佳实践

- [http://b2b-doc.alibaba-inc.com/display/RC/Dubbo Best Practices](http://b2b-doc.alibaba-inc.com/display/RC/Dubbo+Best+Practices)

QA

Q&A