分布式服务管理之dubbox

分布式服务管理之dubbox

系统分析说明书
版本 0.1
项目名称：

修订历史

版本号	作者	修订章节	修订原因	修订日期
0.1	甘建新	初稿	初稿	2016-08-16

目　　录

概述

术语

名称	说明

需求背景

- 当服务越来越多时，服务URL配置管理变得非常困难，Array/F5硬件负载均衡器的单点压力也越来越大。
- 当进一步发展，服务间依赖关系变得错综复杂，不能直观分辨应用启动顺序，不能完整的描述应用间的架构关系。
- 随着服务的调用量越来越大，服务的容量问题就会暴露出来，如何分配系统资源变的非常迫切。

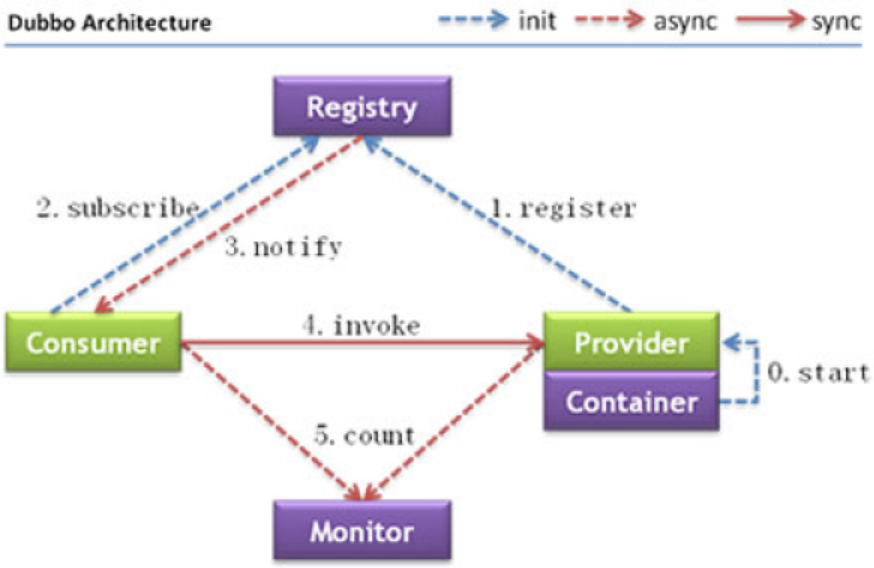
目标

系统目标

- 统一服务注册中心，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Array/F5硬件负载均衡器的依赖，也能减少部分成本。
- 需要自动画出应用间的依赖关系图，以帮助架构师理清应用间依赖关系。
- 需要将服务每天的调用量，响应时间，都统计出来，作为容量规划的参考指标。
- 需要动态调整服务权重，反推服务总容量。

系统分析

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: 统计服务的调用次调和调用时间的监控中心。
Container: 服务运行容器。
调用关系说明：

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。

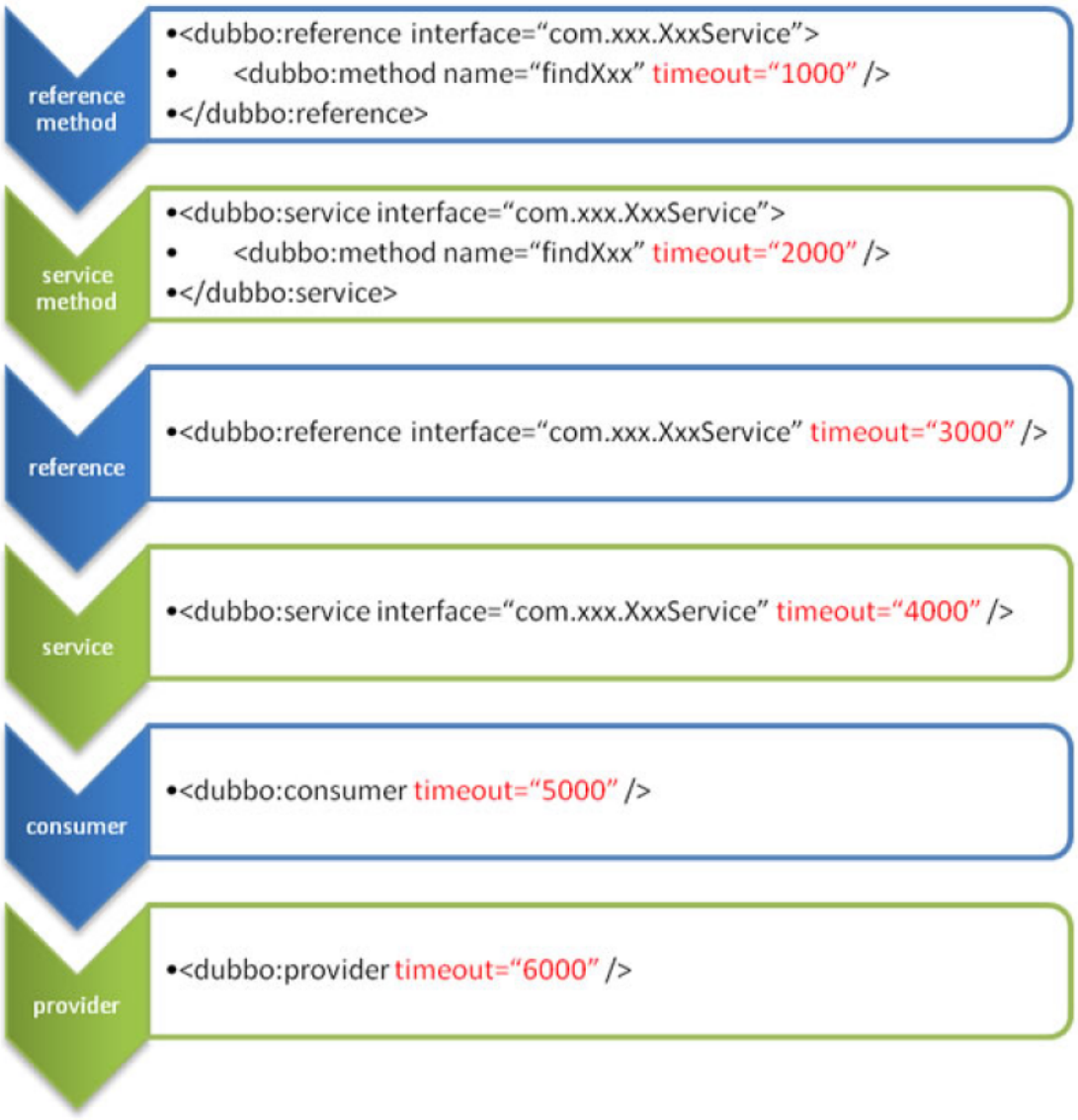
XML配置

所有配置描述

- <dubbo:service/> 服务配置，用于暴露一个服务，定义服务的元信息，一个服务可以用多个协议暴露，一个服务也可以注册到多个注册中心。
- <dubbo:reference/> 引用配置，用于创建一个远程服务代理，一个引用可以指向多个注册中心。

- <dubbo:protocol/> 协议配置，用于配置提供服务的协议信息，协议由提供方指定，消费方被动接受。
- <dubbo:application/> 应用配置，用于配置当前应用信息，不管该应用是提供者还是消费者。
- <dubbo:module/> 模块配置，用于配置当前模块信息，可选。
- <dubbo:registry/> 注册中心配置，用于配置连接注册中心相关信息。
- <dubbo:monitor/> 监控中心配置，用于配置连接监控中心相关信息，可选。
- <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:method/> 方法配置，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。

配置覆盖原则



上图中以timeout为例，显示了配置的查找顺序，其它retries, loadbalance, actives等类似。

- 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。

其中，服务提供方配置，通过URL经由注册中心传递给消费方。
建议由服务提供方设置超时，因为一个方法需要执行多长时间，服务提供方更清楚，如果一个消费方同时引用多个服务，就不需要关心每个服务的超时设置。
理论上ReferenceConfig的非服务标识配置，在ConsumerConfig，ServiceConfig, ProviderConfig均可以缺省配置。

注解配置

- 服务提供方注解：

```
import com.alibaba.dubbo.config.annotation.Service;

@Service(version="1.0.0")
public class FooServiceImpl implements FooService {

    // ......

}
```

- 服务提供方配置：

```
<!-- 公共信息，也可以用dubbo.properties配置 -->
<dubbo:application name="annotation-provider" />
<dubbo:registry address="127.0.0.1:4548" />

<!-- 扫描注解包路径，多个包用逗号分隔，不填pacakge表示扫描当前ApplicationContext
中所有的类 -->
<dubbo:annotation package="com.foo.bar.service" />
```

```
```

- 服务消费方注解：

```
import com.alibaba.dubbo.config.annotation.Reference;
import org.springframework.stereotype.Component;

@Component
public class BarAction {

    @Reference(version="1.0.0")
    private FooService fooService;

}
```

- 服务消费方配置：

```
<!-- 公共信息，也可以用dubbo.properties配置 -->
<dubbo:application name="annotation-consumer" />
<dubbo:registry address="127.0.0.1:4548" />

<!-- 扫描注解包路径，多个包用逗号分隔，不填pacakge表示扫描当前ApplicationContext
中所有的类 -->
<dubbo:annotation package="com.foo.bar.action" />
```

启动检查

Dubbo缺省会在启动时检查依赖的服务是否可用，不可用时会抛出异常，阻止Spring初始化完成，以便上线时，能及早发现问题，默认check=true。

集群容错

在集群调用失败时，Dubbo提供了多种容错方案，缺省为failover重试。

Failover Cluster

- 失败自动切换，当出现失败，重试其它服务器。(缺省)
- 通常用于读操作，但重试会带来更长延迟。
- 可通过retries="0"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。
- 通常用于实时性要求较高的读操作，但需要浪费更多服务资源。
- 可通过forks="2"来设置最大并行数。

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)
- 通常用于通知所有提供者更新缓存或日志等本地资源信息。

负载均衡

在集群负载均衡时，Dubbo提供了多种均衡策略，缺省为random随机调用。

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

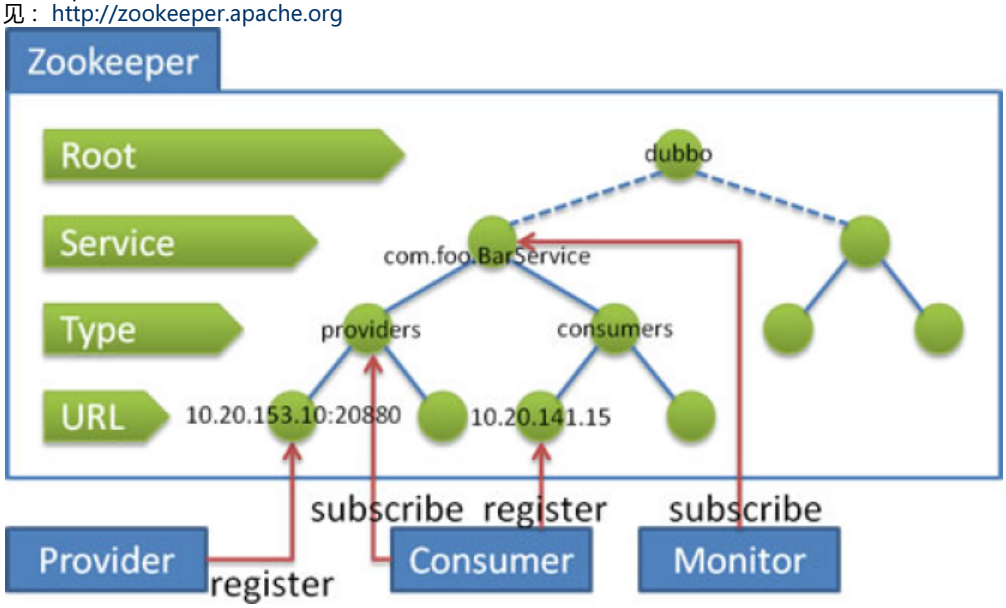
- 一致性Hash，相同参数的请求总是发到同一提供者。
- 当某一台提供者挂时，原本发往该提供者的请求，基于虚拟节点，平摊到其它提供者，不会引起剧烈变动。
- 算法参见： http://en.wikipedia.org/wiki/Consistent_hashing。
- 缺省只对第一个参数Hash，如果要修改，请配置<dubbo:parameter key="hash.arguments" value="0,1" />
- 缺省用160份虚拟节点，如果要修改，请配置<dubbo:parameter key="hash.nodes" value="320" />

参数验证

在目前的改造中不涉及，会安排到后面的计划中。
参数验证功能是基于JSR303实现的，只需标识JSR303标准的验证Annotation，并通过声明filter来实现验证。

注册中心zookeeper

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



流程说明：

- 服务提供者启动时
  - 向/dubbo/com.foo.BarService/providers目录下写入自己的URL地址。
- 服务消费者启动时
  - 订阅/dubbo/com.foo.BarService/providers目录下的提供者URL地址。
  - 并向/dubbo/com.foo.BarService/consumers目录下写入自己的URL地址。
- 监控中心启动时
  - 订阅/dubbo/com.foo.BarService目录下的所有提供者和消费者URL地址。

支持以下功能：

- 当提供者出现断电等异常停机时，注册中心能自动删除提供者信息。
- 当注册中心重启时，能自动恢复注册数据，以及订阅请求。
- 当会话过期时，能自动恢复注册数据，以及订阅请求。
- 当设置<dubbo:registry check="false" />时，记录失败注册和订阅请求，后台定时重试。
- 可通过<dubbo:registry username="admin" password="1234" />设置zookeeper登录信息。
- 可通过<dubbo:registry group="dubbo" />设置zookeeper的根节点，不设置将使用无根树。
- 支持*号通配符<dubbo:reference group="" version="" />，可订阅服务的所有分组和所有版本的提供者。

管理平台

平台地址

开发环境地址 http://192.168.180.43:9000
管理员用户名密码 root/root
访客用户名密码 guest/guest
准生产环境 http://10.255.6.154:9000/
管理员用户名密码 root/root
访客用户名密码 guest/guest
测试环境 http://10.251.6.159:9100/
管理员用户名密码 root/root
访客用户名密码 guest/guest

搜索页面

当你需要管理Dubbo的服务时，首先要搜索到这个服务，然后打开它的管理页面：

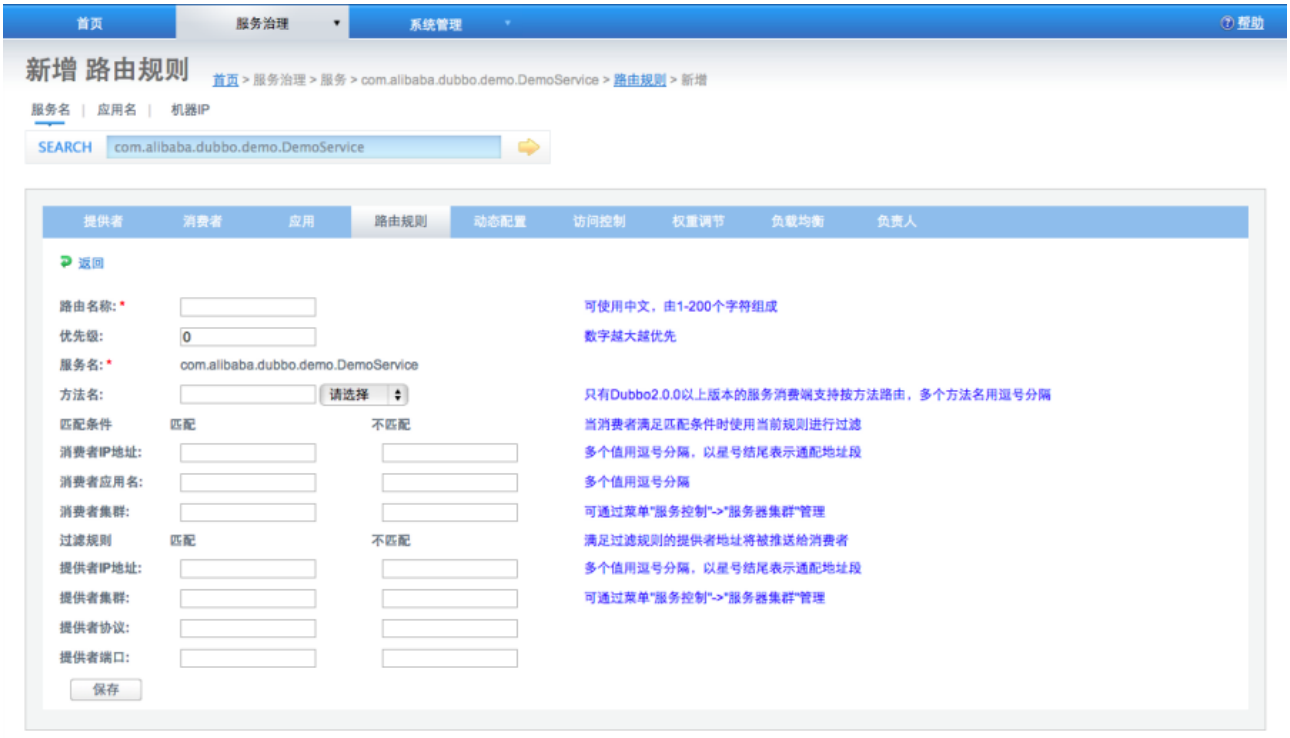
服务提供者页面

服务消费者页面

服务应用页面

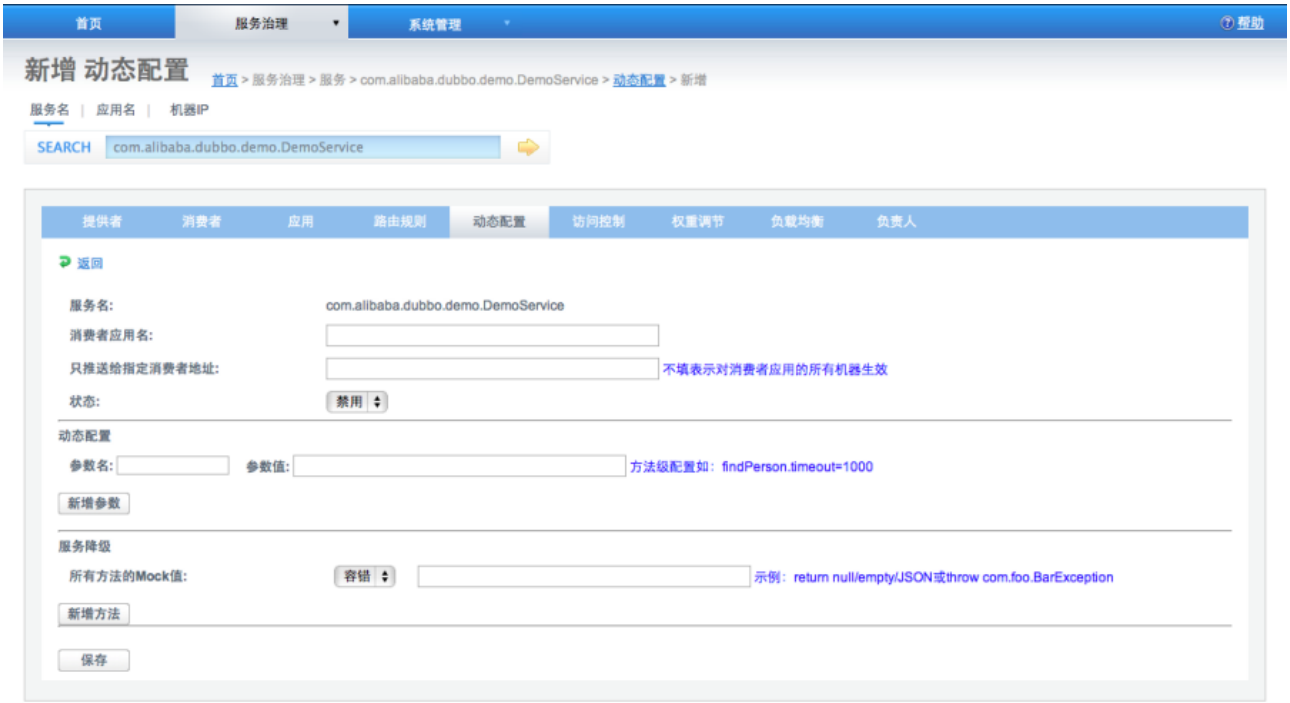

DUBBO
简体中文
root, 您好
退出
首页
服务治理
系统管理
帮助
应用
首页 > 服务治理 > 服务 > com.alibaba.dubbo.demo.DemoService > 应用
服务名 | 应用名 | 机器IP
SEARCH
com.alibaba.dubbo.demo.DemoService
提供者
消费者
应用
路由规则
动态配置
访问控制
权重调节
负载均衡
负责人
批量屏蔽 | 批量容错 | 批量恢复 | 缺省屏蔽 | 缺省容错 | 缺省恢复
应用名:
角色: 所有
降级: 所有
操作
demo-consumer
消费者
未降级
屏蔽 | 容错
demo-provider
提供者
共2条记录

添加路由规则页面


首页
服务治理
系统管理
帮助
新增 路由规则
首页 > 服务治理 > 服务 > com.alibaba.dubbo.demo.DemoService > 路由规则 > 新增
服务名 | 应用名 | 机器IP
SEARCH
com.alibaba.dubbo.demo.DemoService
提供者
消费者
应用
路由规则
动态配置
访问控制
权重调节
负载均衡
负责人
返回
路由名称: *
可使用中文，由1-200个字符组成
优先级:
0
数字越大越优先
服务名: *
com.alibaba.dubbo.demo.DemoService
方法名:
请选择
只有Dubbo2.0.0以上版本的服务消费端支持按方法路由，多个方法名用逗号分隔
匹配条件
匹配
不匹配
当消费者满足匹配条件时使用当前规则进行过滤
消费者IP地址:
多个值用逗号分隔，以星号结尾表示通配地址段
消费者应用名:
多个值用逗号分隔
消费者集群:
可通过菜单"服务控制"->"服务器集群"管理
过滤规则
匹配
不匹配
满足过滤规则的提供者地址将被推送给消费者
提供者IP地址:
多个值用逗号分隔，以星号结尾表示通配地址段
提供者集群:
可通过菜单"服务控制"->"服务器集群"管理
提供者协议:
提供者端口:
保存

在日常开发过程中，如果本地起两个应用，希望本地访问本地，可通过路由规则配置。

添加动态配置页面

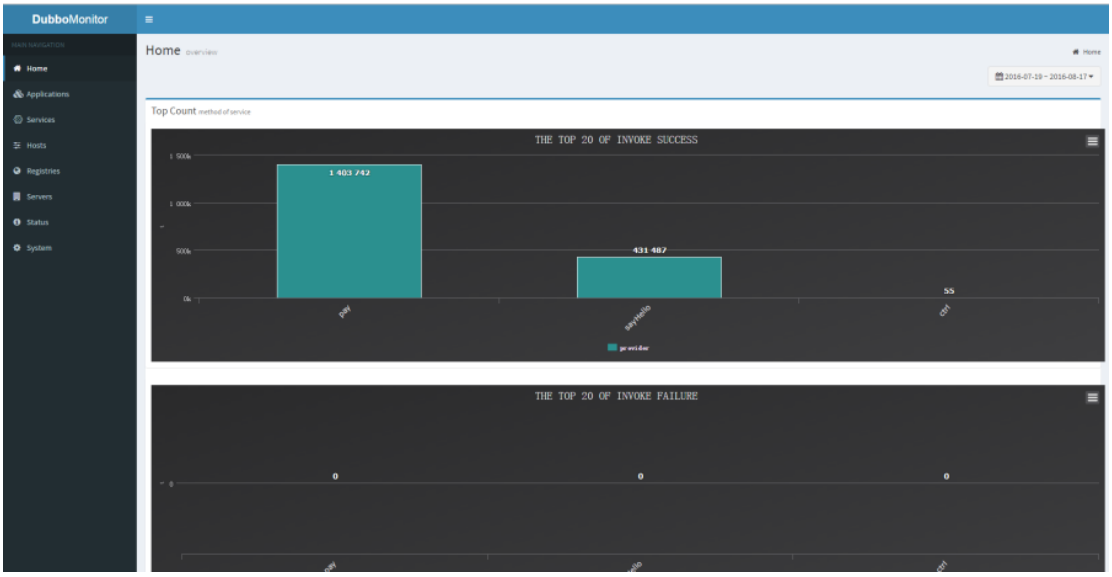


监控平台

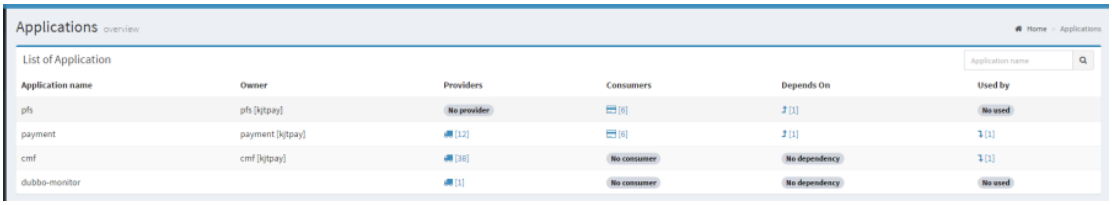
平台地址

开发环境地址 http://192.168.180.43:9002
用户名密码 admin/admin
准生产环境 http://10.255.6.154:9002
用户名密码 admin/admin
测试环境 http://10.251.6.159:9102
用户名密码 admin/admin

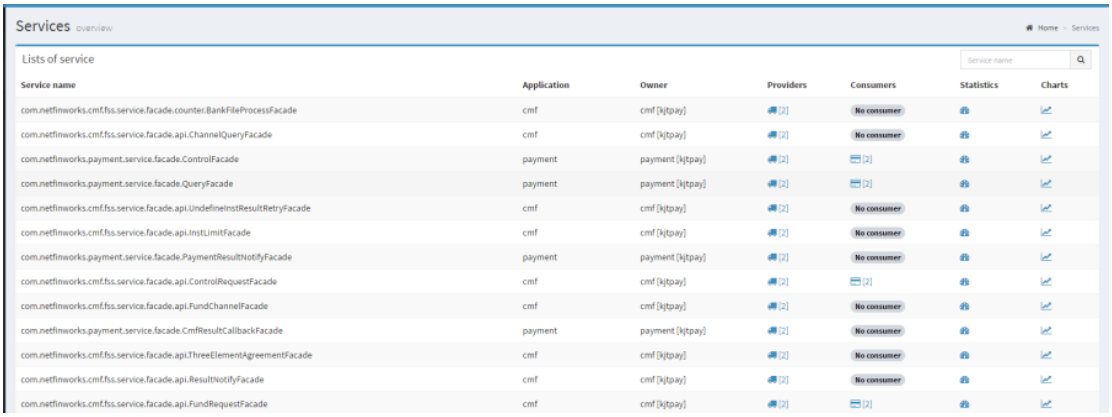
首页

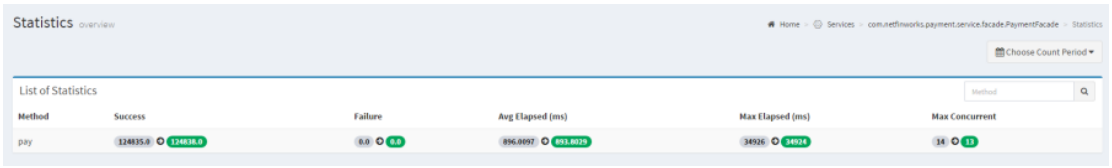


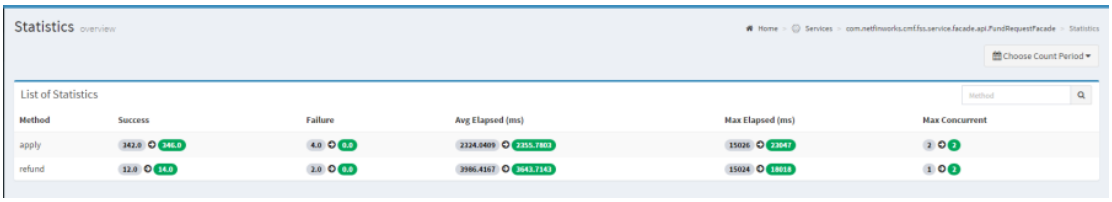
应用列表



Service统计







dubbo与dubbox区别与联系

dubbox是当当网开源的项目，为Dubbo服务框架提供多项扩展功能。
新功能包括：

- 支持REST风格远程调用（HTTP + JSON/XML)
- 支持基于Kryo和FST的Java高效序列化实现
- 支持基于嵌入式Tomcat的HTTP remoting体系
- 升级Spring
- 升级ZooKeeper客户端

代码修改步骤

以下配置基于payment工程描述，各项目负责人基于该配置进行修改。由于当前所有工程都采用xml配置方式，在本次更换dubbox架构的过程中依旧采用xml配置的方式进行修改。

Jar包依赖

需要在项目父工程中pom.xml添加jar依赖。
在properties节点添加以下内容：

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

在dependencies节点中添加以下内容：

```
<!-- 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>
```

属性文件

文件名称：dubbo.properties
SVN地址：http://192.168.180.67/svn/src/pay/config/basis/payment/dubbo.properties
详细内容如下：

```
############################################################ 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
## 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
############################################################ 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=200
## 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
dubbo.consumer.retries=0
```

Spring公共配置文件

在web工程中添加application-dubbo.xml文件

```
<?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="${dubbo.application.name}" version="1.0.0"
owner="${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}" />
<!-- 监控 -->
<dubbo:monitor protocol="registry" />
</beans>
```

在spring主配置文件application.xml引入新加的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" />
```

生产者配置文件

在payment-ext-service项目中添加生产者配置文件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>
```

消费者配置文件

在payment-ext-integration项目中添加消费者配置文件dubbo-consumer.xml，目前服务并发量不是特别大，在此处负载均衡方式设置为轮询roundrobin

```
<?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>
```

注册中心地址

所有环境之中不同之处在于zookeeper地址不同。所需修改文件为dubbo.properties

开发环境

```
## 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
```

准生产环境

```
## 注册中心地址
dubbo.registry.address=zookeeper://10.255.6.154:2181?backup=10.255.6.154:2181,10.255.6.154:2181
```

测试环境

```
## 注册中心地址
dubbo.registry.address=zookeeper://10.251.6.159:2181?backup=10.251.6.159:2182,10.251.6.159:2183
```

生产环境

```
## 注册中心地址
dubbo.registry.address=zookeeper://zookeeper1.kjtpay.com:2181?backup=zookeeper2.kjtpay.com:2181,zookeeper3.kjtpay.com:2181
```

容灾环境

同生产环境配置。

下一步计划

生产环境	Pfs,payment
准生产环境	Pfs,payment,cmf

下一步修改渠道，交易。

参考资料

Dubbo官网: http://dubbo.io/
Zookeeper官网: http://zookeeper.apache.org
Dubbo Git地址：https://github.com/alibaba/dubbo
Dubbox Git地址：https://github.com/dangdangdotcom/dubbox
Dubbox SVN源码：http://192.168.180.67/svn/src/basis/dubbox/branches/dubbo-parent
注：当前使用的dubbox版本是SVN地址编译而来，与git有部分jar版本和参数区别。