# Dubbo

## 基础对象

### Dubbo服务端最核心最基础的四个对象

特点 : 配置 , 暴露

Applicationconfig---------提供名字

Protocolconfig------------提供协议

Registerconfig-------------注册中心

Serviceconfig--------------整理包含其他配置

ApplicationConfig applicationConfig = new ApplicationConfig("sample-app");  
ProtocolConfig protocolConfig = new ProtocolConfig();  
protocolConfig.setName("dubbo");  
protocolConfig.setPort(-1);*//dubbo默认端口号20880  
//不需要注册中心*RegistryConfig registryConfig = new RegistryConfig(RegistryConfig.*NO\_AVAILABLE*);  
ServiceConfig serviceConfig = new ServiceConfig();  
serviceConfig.setInterface(UserService.class);  
serviceConfig.setRef(new UserServiceImpl());  
serviceConfig.setApplication(applicationConfig);  
serviceConfig.setProtocol(protocolConfig);  
serviceConfig.setRegistry(registryConfig);  
*//服务暴露*serviceConfig.export();

### 客户端最基础俩个对象

特点 : 配置 , 引用

Applicationconfig-------------提供名字

Referenceconfig--------------整合其他信息

ApplicationConfig applicationConfig = new ApplicationConfig("young-app");  
ReferenceConfig referenceConfig = new ReferenceConfig();  
referenceConfig.setApplication(applicationConfig);  
referenceConfig.setInterface(UserService.class);  
referenceConfig.setUrl("dubbo://192.168.56.1:20880/com.majian.client.service.UserService");  
UserService userService = (UserService) referenceConfig.get();

## spring boot依赖



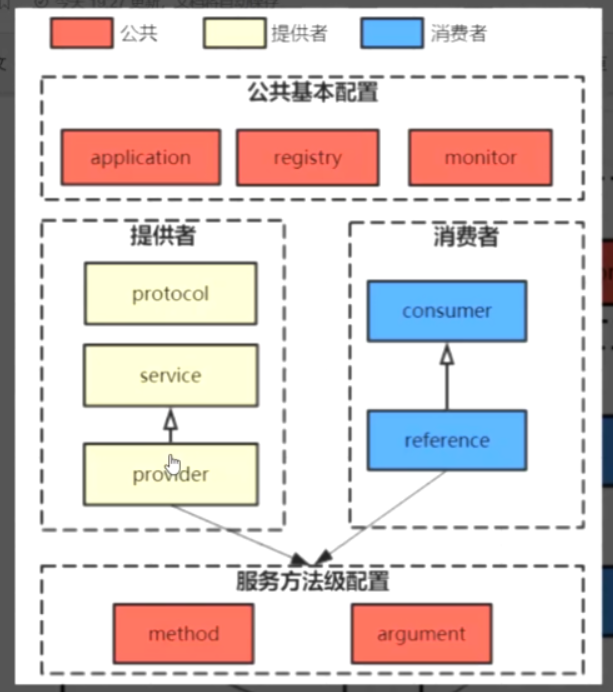
## springboot使用

在接口实现类上面发放两个注解

@Service

@Component

在application上面加@EnableDubbo



Provider是service的模板，一般在项目中有多个service，，但是只有一个provider

Service的优先级大于provider

Consumer是reference的模板

Monitor 监控中心

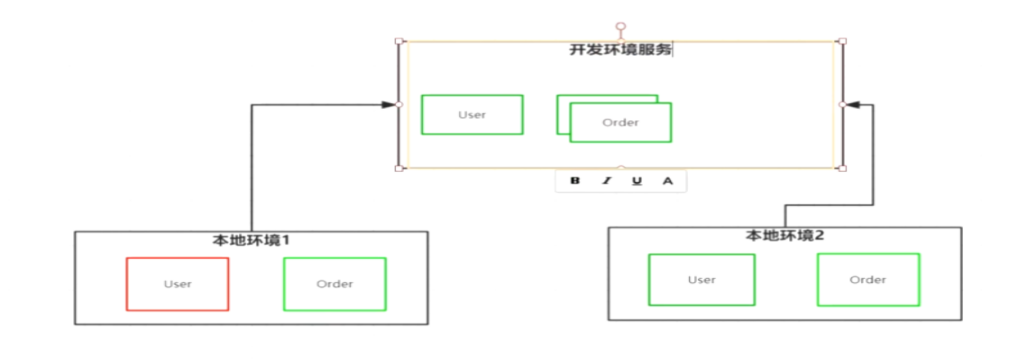
## Ssm配置

### 服务端配置



Register=true/flase,flase不暴露端口

不暴露服务，用于防止别的本地服务连接到自己的服务，相当于我只对开发环境服务进行引用，不进行服务提供

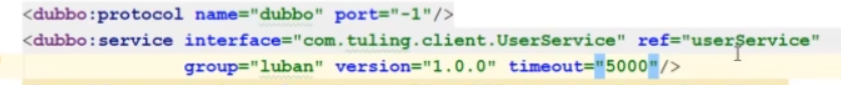




Subscribe = true 指是否订阅，默认true

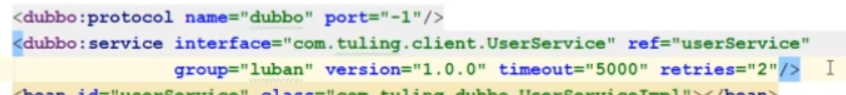


忽略注册中心没有 这个错误，在环境启动方式后不按照顺序启动，check=false 忽略错误，然后会重连



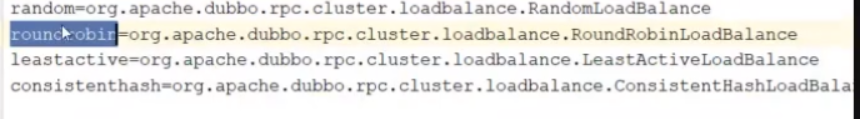
Service 接口\

Timeout 超时时间



Retries 重连次数，，实际是三次，，只有失败了才会重新连接，异常情况下不会重试，超时情况下会重试





轮询策略

Loadbalance = 有四种

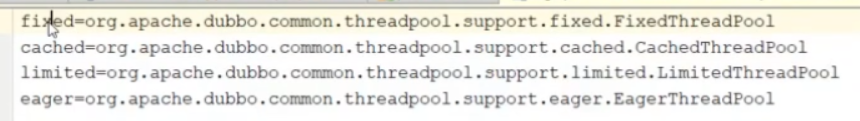


Provider相当于模板，service中都会默认配置，除非覆盖



Threads 线程数量

Threadpool 线程池模型



线程池模型有四种

### 客户端配置

一个服务既可以是消费者，也可以是服务者



虽然在服务端配置，但是用在客户端，出异常了在客户端报异常

例如：timeout 超时时间， retries 重试次数 loadbalance 轮询规则

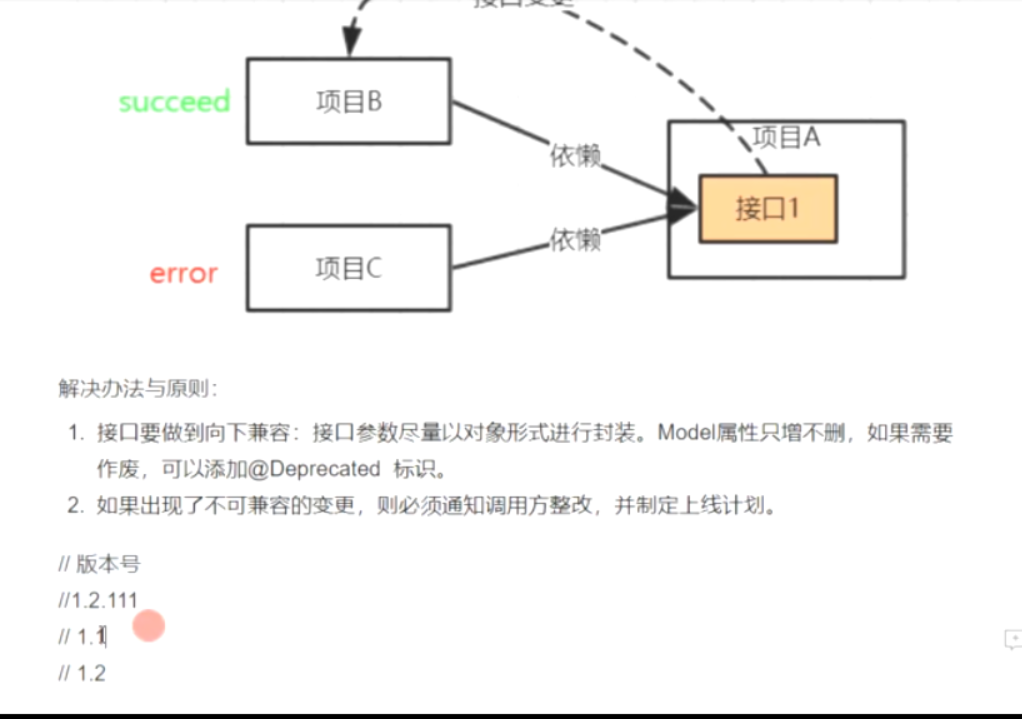
如果把timeout，retries等信息配置在服务端，服务端更可以给客户端一个好的建议，但是如果服务端客户端都配置了，那么还是要遵循客户端的，因为dubbo作者的意图是 服务提供者更加的明白这个服务怎么去调，到底需要多长时间，

可以配置timeout的地方如下----6个地方，，也可以是7个



在dubbo-admin中也可以设置，，，daubbo-admin的优先级最高

当有接口变更，会出现不兼容错误



传的参数尽量不要是map，object，json等不清晰的参数





如果客户端和服务端都配置了分组，那么只有在一个组的才可以调用



## Dubbo----admin



Dubbo默认为随机，可以加权重

倍权，就是权重加大



在这里就可以看到权重



所有ip符合192.168.0.150的消费者在调用getUser方法时，只能调用端口号为20880的服务者

目录

## 注册中心

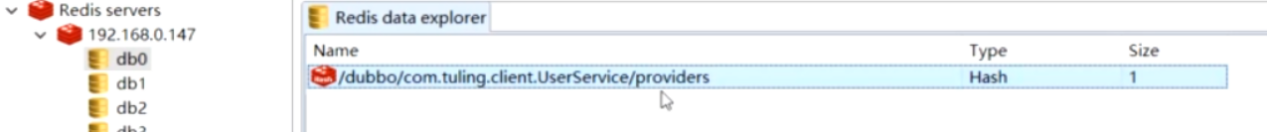
### Redis

#### 服务端

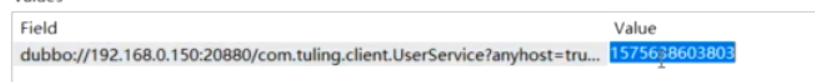
Redis有订阅发布的功能

当有服务注册时，会存储一个hash类型值

Key为/dubbo/加上接口全限定类名/家注册者为providers，



Field为地址加接口加程序 和 value为服务有效期



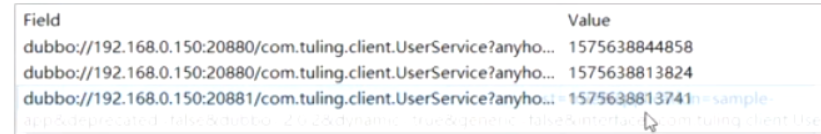
发送一次心跳，来续命60秒



这个线程默认每隔30秒钟执行一次

客户端会通过value来判断是否过期

服务方节点停止，value也不会被删除，就是没有线程给他续命，就会过期



当服务方重新启动，他不会覆盖失效的之前的，会重新生成一个hash的数据

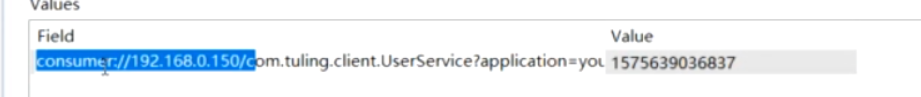
客户端和服务端之间也有长连接（心跳机制），两秒发生一次，不会因为redis注册中心value有效期为60秒为产生大麻烦（这是因为服务端挂掉后value数据就没有线程去修改了，redis就不能感知到，如果手动修改value数据，那么注册中心redis会立马感知到）

#### 客户端



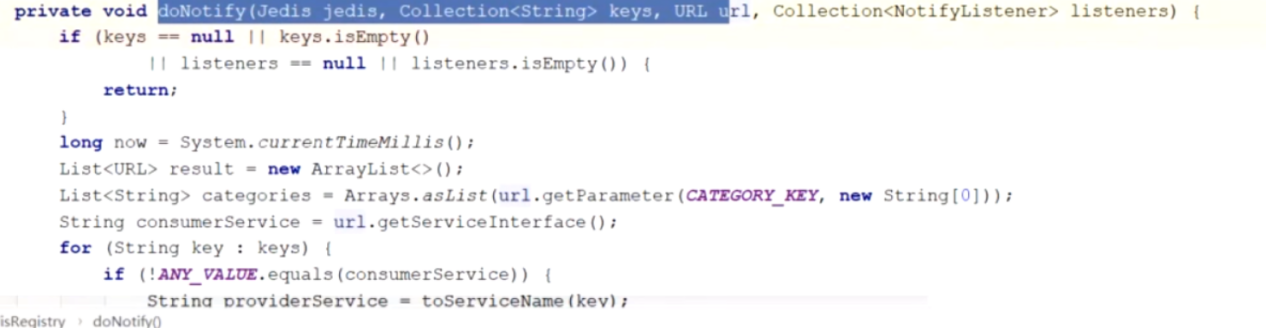
前面和服务端一样，后面是consumers

存储的也是hash类型数据



#### Redis源码

如果redis数据有变更，就会触发doNotify方法

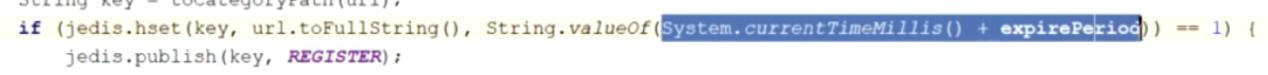


这个线程专门用来监听redis数据的变更·



当节点数据发生变更，首先会获取key值，通过key值获取变更field和value，然后查看value是否有效，只有有效才会加载进来

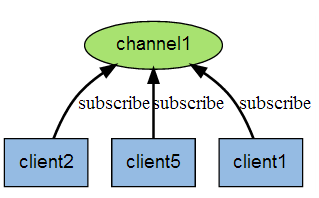
心跳机制的续命方法



定时设置，每30秒加载一次，当前时间加60秒钟，然后保存进去

#### 总结：

Redis注册中心是通过pub和sub来进行订阅发布



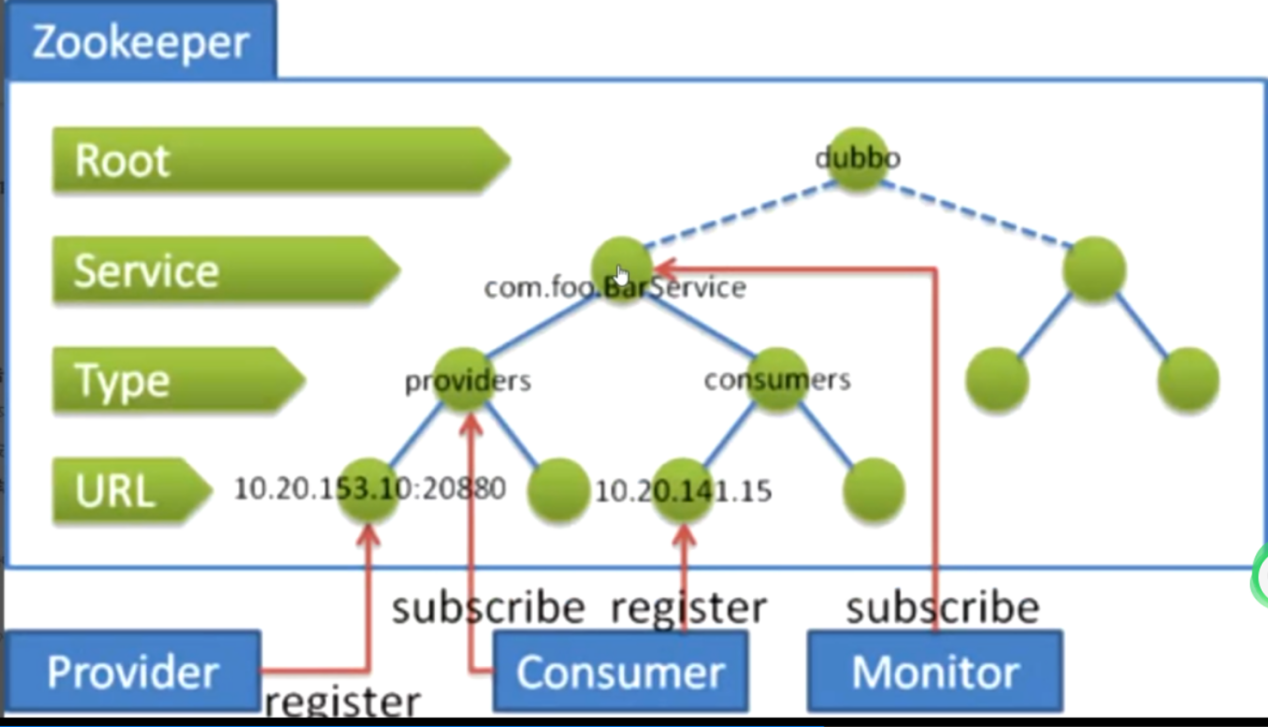
当有新消息通过 PUBLISH 命令发送给频道 channel1 时， 这个消息就会被发送给订阅它的三个客户端：

首先服务端启动会在redis上面存储一个hash类型的值，key为dubbo加接口的全限定类型，field为dubbo协议加上ip加上接口全限定类名加上程序等一堆，value为有效时间，第一次有效时间为当前时间加60秒，然后服务端有一个专门的线程每隔60秒/2,也就是30秒钟会有一此给value续命的方法，每一次都会把value值修改成当前时间加60秒来不让value不过期，

客户端启动也会在redis上面存储hash类型数据，和服务端结构差不多，客户端会有专门的线程去监控redis上面的数据是否会发生改变，如果发生改变立即就会先获取更改的key值，然后在获取到field和value，然后查看value是否过期，如果过期，则不会把这个服务加载进来，客户端和服务端之间有长连接，类似于心跳机制2秒一次，如果服务端挂掉后客户端会立刻感知到，然后不调用这个宕机的服务接口，这样就不会因为挂掉的服务端有60秒的存活期发生异常。，当挂掉的服务端重启，不会覆盖之前的value，而是会重新存储一个新的value，旧的value不会被删除。

Redis注册中心不能立刻感知到服务端挂掉就是因为有60秒的存活期。

### Zookeeper



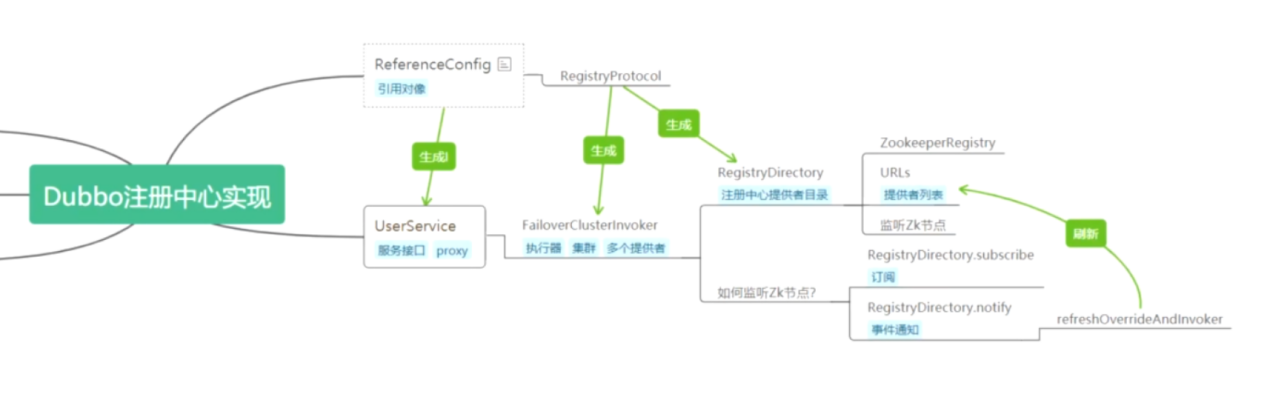
Url为临时节点，前面的都是持久节点， configurators下面为持久节点

在zookeeperi中，当有服务端和客户端时



节点下会有consumers客户端，provters服务端，configurators配置，就是dubbo-admin中配置的配置

调用原理



## Dubbo调用原理

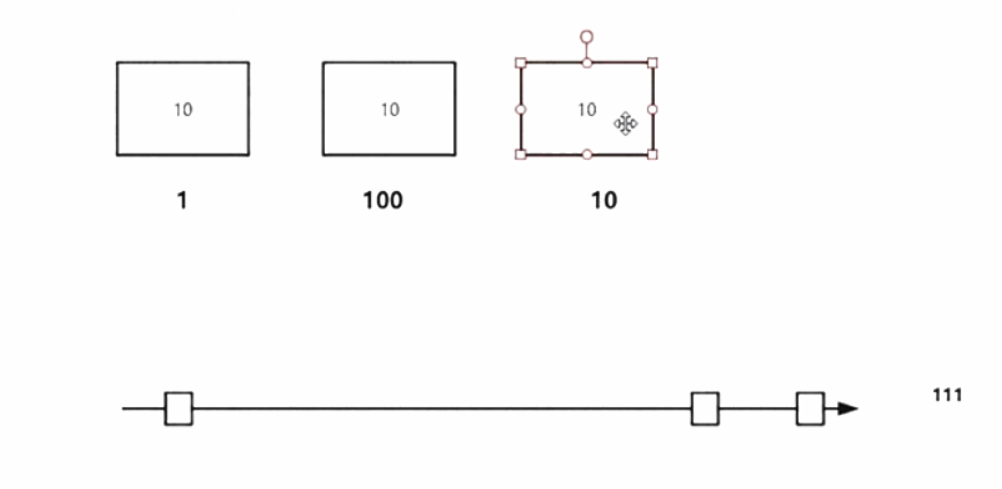
### 负载均衡算法： ---客户端用的

权重随机：按照权重设置随机概率，默认算法

权重轮询：按照权重设置轮询占比

最少活跃数调用数：相同活跃数的权重随机，活跃数为前后调用次数差

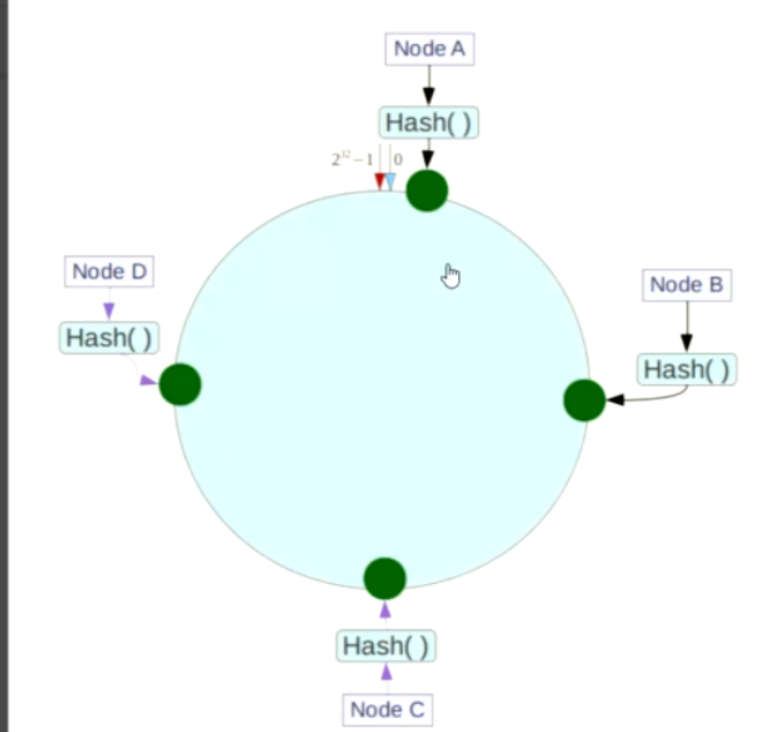
当调用次数有差别时，直接调用调用次数最少的，如果调用次数相同，会很具权重，来进行权重随机来调用，不一定调用权重高的服务



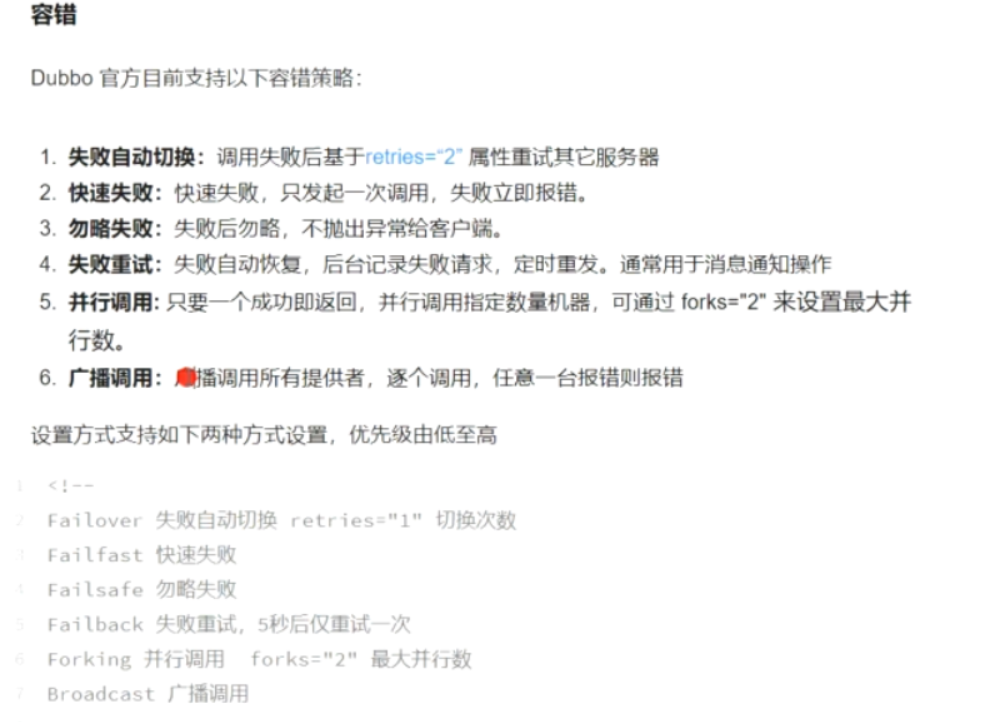
假设有三个服务提供方，权重分别为1 100 10，总权重为111，这时算法会随机生成一个0到111之间的数，最后看落到哪个区间，就调用谁

一致性哈希: 相同参数总是发送到同一台机器

对服务提供者的IP加端口号进行哈希运算取值，最后得出一个值，对应着一个服务提供着



## 容错

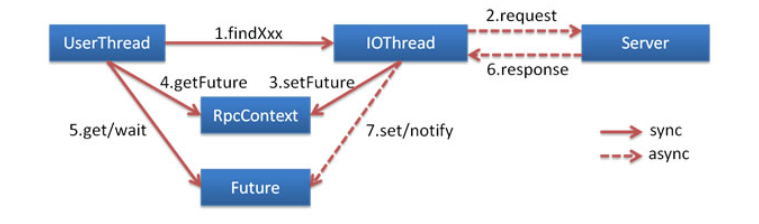


同步异步调用

异步调用是指发起远程调用之后获取结果的方式。

1. 同步等待结果返回（默认）
2. 异步等待结果返回
3. 不需要返回结果

Dubbo 中关于异步等待结果返回的实现流程如下图：



异步调用配置:

<dubbo:reference id="asyncDemoService"

                 interface="com.tuling.teach.service.async.AsyncDemoService">

                 <!-- 异步调async：true 异步调用 false 同步调用-->

    <dubbo:method name="sayHello1" async="false"/>

    <dubbo:method name="sayHello2" async="false"/>

     <dubbo:method name="notReturn" return="false"/>

</dubbo:reference>