# **springboot dubbo负载均衡配置**



[有丶意思](https://me.csdn.net/qq_30591163" \t "https://blog.csdn.net/qq_30591163/article/details/_blank) 2019-05-28 10:20:14 IMG_257 1968 IMG_258 收藏 6

分类专栏： [springboot](https://blog.csdn.net/qq_30591163/category_7945891.html" \t "https://blog.csdn.net/qq_30591163/article/details/_blank) [dubbo](https://blog.csdn.net/qq_30591163/category_7948823.html" \t "https://blog.csdn.net/qq_30591163/article/details/_blank) 文章标签： [dubbo](https://www.csdn.net/gather_2d/MtTaEg0sMzU0NDEtYmxvZwO0O0OO0O0O.html" \t "https://blog.csdn.net/qq_30591163/article/details/_blank) [springboot dubbo负载均衡配置](https://so.csdn.net/so/search/s.do?q=springboot dubbo%E8%B4%9F%E8%BD%BD%E5%9D%87%E8%A1%A1%E9%85%8D%E7%BD%AE&t=blog&o=vip&s=&l=&f=&viparticle=" \t "https://blog.csdn.net/qq_30591163/article/details/_blank)

版权

### **springboot dubbo负载均衡配置**

* + [1、多zookeeper yml配置](https://blog.csdn.net/qq_30591163/article/details/90634169" \l "1zookeeper_yml_1" \t "https://blog.csdn.net/qq_30591163/article/details/_self)
  + [2、负载配置](https://blog.csdn.net/qq_30591163/article/details/90634169" \l "2_6" \t "https://blog.csdn.net/qq_30591163/article/details/_self)
  + [random](https://blog.csdn.net/qq_30591163/article/details/90634169" \l "random_11" \t "https://blog.csdn.net/qq_30591163/article/details/_self)
  + [roundRobin](https://blog.csdn.net/qq_30591163/article/details/90634169" \l "roundRobin_14" \t "https://blog.csdn.net/qq_30591163/article/details/_self)
  + [leastActive](https://blog.csdn.net/qq_30591163/article/details/90634169" \l "leastActive_17" \t "https://blog.csdn.net/qq_30591163/article/details/_self)
  + [consistentHash](https://blog.csdn.net/qq_30591163/article/details/90634169" \l "consistentHash_20" \t "https://blog.csdn.net/qq_30591163/article/details/_self)

## **1、多zookeeper yml配置**

dubbo:

registry:

protocol: zookeeper

address: 127.0.0.1:2181|192.168.34.109:2181

* 4

## **2、负载配置**

dubbo:

reference:

loadbalance: roundrobin

* 3

## **random**

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。

* 2

## **roundRobin**

轮询，按公约后的权重设置轮询比率。

存在慢的提供者类即请求问题，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。

* 2

## **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。

* 2

## **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，平摊到其他提供者，不会引起剧烈变动。

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

### **1. 服务端服务级别**

****1.1 XML配置方式：****

<dubbo:service interface="..." loadbalance="roundrobin" />

****1.2 yml配置（SpringBoot）:****

dubbo:

provider:

loadbalance: roundrobin

****1.3 基于注解的配置（SpringBoot）：****

具体可以查看dubbo注解：@Service

@Service(version = "${product.service.version}",loadbalance="roundrobin")

public class ProductServiceImpl implements ProductService {

@Override

public String helloDubbo() {

return "hello,spring-boot dubbo";

}

}

### **2. 客户端服务级别**

****2.1 XML配置方式：****

<dubbo:reference interface="..." loadbalance="roundrobin" />

****2.2 yml配置（SpringBoot）****

dubbo:

consumer:

loadbalance: roundrobin

****2.3 基于注解的配置（SpringBoot）：****

   具体可以查看注解：@Reference

@RestController

public class ProductController {

@Reference(version = "${product.service.version}",loadbalance="roundrobin")

private ProductService productService;

@RequestMapping(value = "test")

public String test() {

return productService.helloDubbo();

}

**我们也可以通过可视化界面管理平台来操作**

