每周一问：微服务如何实现负载均衡？

一、内容提要

1. 为什么使用负载均衡
2. SpringCloud负载均衡的技术组件？
3. Ribbon负载均衡的几种方式
4. 自定义负载均衡算法
5. 主要内容

1.为什么使用负载均衡？

随着网站、应用访问量的增加，一台服务器已经不能满足应用的需求，而需要多台服务器集群，这时就会用到负载均衡。

它的好处：

1).负载均衡优化了访问请求在服务器组之间的分配，消除了服务器之间的负载不平衡，从而提高了系统的反应速度与总体性能；

2).负载均衡可以对服务器的运行状况进行监控，及时发现运行异常的服务器，并将访问请求转移到其它可以正常工作的服务器上，从而提高服务器组的可靠性采用了负均衡器器以后，可以根据业务量的发展情况灵活增加服务器，系统的扩展能力得到提高，同时简化了管理。

1. SpringCloud负载均衡的技术组件（以Rinbbon为例）

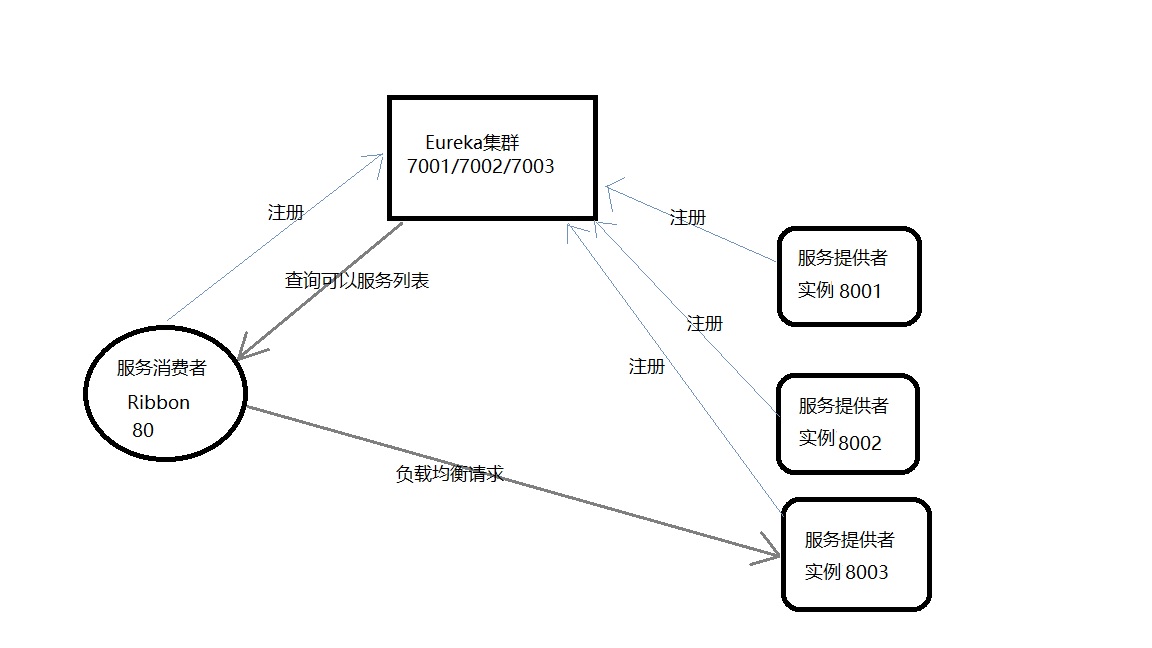
负载均衡分为硬件负载均衡和软件负载均衡，微服务中的负载均衡是软负载的一种，简单灵活，易于扩展。Rinbbon就是SpringCloud中比较流行的一种负载均衡组件。

Ribbon负载均衡机制介绍：

Ribbon是Netflix公司开源的一个客户端负载均衡的项目，可以自动与 Eureka 进行交互。

1. 负载均衡代码实现

项目架构图



项目准备：

上一期每周一问已经搭建好Eureka集群，

简单数据准备：

创建三个数据库及表cloudDB01/cloudDB02/cloudDB03

创建三个服务提供者8001/8002/8003并注册到Eureka集群中，

分别连接对应数据库cloudDB01/cloudDB02/cloudDB03

注意：主启动类注明是Eureka客户端@EnableEurekaClient。

搭建Ribbon服务：

1.修改consumer-dept-80 的pom文件添加信息

Ribbon需要跟Eureka整合

<!-- Ribbon相关 -->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-ribbon</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

2.修改consumer-dept-80 yml，添加信息

eureka:

client: #客户端注册进eureka服务列表内

register-with-eureka: **false** #排除自己不注册

service-url:

#defaultZone: http://localhost:7001/eureka

defaultZone: http://eureka7001.com:7001/eureka/,http://eureka7002.com:7002/eureka/,http://eureka7003.com:7003/eureka/

3.在consumer-dept-80工程的ConfigBean类 getRestTemplate方法添加注解

@LoadBalanced

ConfigBean.java

4.在consumer-dept-80 启动类添加注解@EnableEurekaClient

5. consumer-dept-80 修改DeptController\_Consumer

微服务名称（这样才符合规定）：

6. 测试

依次启动Eureka 7001 7002 7003

再启动3个Dept-8001 8002 8003，各自测试

再启动80，测试负载均衡、轮询策略

localhost/consumer/dept/list

结果显示会不断切换访问各自数据库信息

http://localhost/consumer/dept/get/1

Ribbon工作分为两步：

第一步先选择Eureka Server，它优先选择在同一个区域负载较少的server；

第二步再根据用户指定的策略，在从server取到的服务注册列表中选择一个地址。

其中Ribbon提供了多种策略：比如轮询、随机和根据响应时间加权

Ribbon负载均衡几种策略？

1. **RoundRobinRule轮询（默认）**

第一次到A，第二次就到B，第三次又到A，第四次又到B…  
具体实现是一个负载均衡算法：第N次请求 % 服务器集群的总数 = 实际调用服务器位置的下标  
那么怎么保证线程安全问题呢？因为N次请求次数会自增，怎么保证不会多次请求都拿到同一个N进行自增？答案就是简单的CAS

1. **RandomRule随机**
2. **RetryRule轮询重试（重试采用的默认也是轮询）**

**先按照RoundRobinRule策略获取服务，如果获取的服务失败则在指定的时间内会进行重试，获取可用的服务**

1. **WeightedResponseTimeRule响应速度决定权重：**

对RoundRobbin的一种增强，加入了权重和计算响应时间的概念，其中响应速度最快的权重越大，权重越大则选中的概率越大。

1. **BestAvailableRule最优可用（底层也有RoundRobinRule）：**

最优可用，会过滤掉由于多次访问故障而处于断路器跳闸状态的服务，然后选择一个并发量较小的服务。

1. **AvailabilityFilteringRule可用性过滤规则（底层也有RoundRobinRule）：**

直接翻译就是可用过滤规则，其实它功能是先过滤掉不可用的Server实例，再选择并发连接最小的实例。

1. **ZoneAvoidanceRule区域内可用性能最优**

基于AvailabilityFilteringRule基础上做的，首先判断一个zone的运行性能是否可用，剔除不可用的区域zone的所有server，然后再利用AvailabilityPredicate过滤并发连接过多的server。

8)自定义负载均衡算法

RandomRule\_Adam.java

源码刨析

[Ribbon源码解析 - 方志朋的博客 (fangzhipeng.com)](https://www.fangzhipeng.com/springcloud/2017/08/11/Ribbon-resources.html)

## 总结

综上所述，Ribbon的负载均衡，主要通过LoadBalancerClient来实现的，而LoadBalancerClient具体交给了ILoadBalancer来处理，ILoadBalancer通过配置IRule、IPing等信息，并向EurekaClient获取注册列表的信息，并默认10秒一次向EurekaClient发送“ping”,进而检查是否更新服务列表，最后，得到注册列表后，ILoadBalancer根据IRule的策略进行负载均衡。

而RestTemplate 被@LoadBalance注解后，能过用负载均衡，主要是维护了一个被@LoadBalance注解的RestTemplate列表，并给列表中的RestTemplate添加拦截器，进而交给负载均衡器去处理。

1. QA

集群项目中，除了Ribbon技术组件可以实现负载均衡，还有哪些技术可以实现，各有什么优缺点？列举一二

负载均衡几种机制：

<https://blog.csdn.net/whiteBearClimb/article/details/108703356>

源码地址

[ribbon/ribbon-loadbalancer/src/main/java/com/netflix/loadbalancer at master · Netflix/ribbon (github.com)](https://github.com/Netflix/ribbon/tree/master/ribbon-loadbalancer/src/main/java/com/netflix/loadbalancer)