# Spring Cloud答案

# 简述题：负载均衡的技术分类

负载均衡一般分为服务器端负载均衡和客户端负载均衡

所谓服务器端负载均衡，比如Nginx、F5这些，请求到达服务器之后由这些负载均衡器根据一定的算法将请求路由到目标服务器处理。

所谓客户端负载均衡，比如我们要说的Ribbon，服务消费者客户端会有一个服务器地址列表，调用方在请求前通过一定的负载均衡算法选择一个服务器进行访问，负载均衡算法的执行是在请求客户端进行。

Ribbon是Netflix发布的负载均衡器。Eureka一般配合Ribbon进行使用，Ribbon利用从Eureka中读取到服务信息，在调用服务提供者提供的服务时，会根据一定的算法进行负载。

# 面试题：Ribbon支持的负载均衡算法有哪些

| **负载均衡策略** | **描述** |
| --- | --- |
| RoundRobinRule：轮询策略 | 默认超过10次获取到的server都不可用，会返回一个空的server |
| RandomRule：随机策略 | 如果随机到的server为null或者不可用的话，会while不停的循环选取 |
| RetryRule：重试策略 | 一定时限内循环重试。默认继承RoundRobinRule，也支持自定义注入，RetryRule会在每次选取之后，对选举的server进行判断，是否为null，是否alive，并且在500ms内会不停的选取判断。而RoundRobinRule失效的策略是超过10次，RandomRule是没有失效时间的概念，只要serverList没都挂。 |
| BestAvailableRule：最小连接数策略 | 遍历serverList，选取出可用的且连接数最小的一个server。该算法里面有一个LoadBalancerStats的成员变量，会存储所有server的运行状况和连接数。如果选取到的server为null，那么会调用RoundRobinRule重新选取。 |
| AvailabilityFilteringRule：可用过滤策略 | 扩展了轮询策略，会先通过默认的轮询选取一个server，再去判断该server是否超时可用，当前连接数是否超限，都成功再返回。 |
| ZoneAvoidanceRule：区域权衡策略**（默认策略）** | 扩展了轮询策略，继承了2个过滤器：ZoneAvoidancePredicate和AvailabilityPredicate，除了过滤超时和链接数过多的server，还会过滤掉不符合要求的zone区域里面的所有节点， 在一个区域/机房内的服务实例中轮询。**先过滤再轮询** |

# 面试题：Hystrix的舱壁模式

Hystrix默认情况下是共享线程池，适用于服务调用比较小的情况。

调用默认是使用相同的线程来执行调用的，这些线程Java容器为处理所有请求预留的。在高服务器请求的情况下，一个性能较低的服务会“霸占”java容器中绝大多数线程，而其它性能正常的服务的请求则需要等待线程资源的释放。最后，整个java容器会崩溃。舱壁模式能将远程调用隔离在各个远程调用自己的线程池中，因此单个性能出问题的服务能得到控制，java容器也不会崩溃。

Hystrix将远程服务的请求托管在一个线程池中。即默认情况下，所有Hystrix命令(@HystrixCommand)共享同一个线程池来处理这些请求。该线程池中持有10个线程来处理各种远程服务请求，可以是REST服务调用、数据库访问等。

为了避免问题服务请求过多导致正常服务无法访问，Hystrix 不是采用增加线程数，而是单独的为每一个控制方法创建一个线程池的方式，这种模式叫做“舱壁模式"，也是线程隔离的手段。