分布式服务架构的功能特性

服务订阅发布

配置化发布和引用服务

服务自动发现机制

服务在线注册和去注册

服务路由

集群容错

服务调用

多协议

序列化方式

统一配置

Spring Cloud 各个组件介绍

Spring Cloud Config

Spring Cloud Netflix

Service Discovery: Eureka Clients

Service Discovery: Eureka Server

Circuit Breaker: Hystrix Clients

服务熔断



服务熔断和电路熔断是一个道理，如果一条线路电压过高，保险丝会熔断，防止出现火灾，但是过后重启仍然是可用的。

而服务熔断则是对于目标服务的请求和调用大量超时或失败，这时应该熔断该服务的所有调用，并且对于后续调用应直接返回，从而快速释放资源，确保在目标服务不可用的这段时间内，所有对它的调用都是立即返回，不会阻塞的。再等到目标服务好转后进行接口恢复。

客户端在调用服务端之前加上熔断控制器，进行目标服务的调用，如果在规定的次数内都为成功，则认为该服务不可用，对于该服务的所有调用返回一个容错值，并设置一个定时线程，在一定的时间后重新恢复服务。

与服务降级有什么区别呢？

比如一个业务场景的服务调用链中调用多个服务，有些服务不是该业务场景的必须服务，则可以进行降级处理。服务降级，就是这么回事，整体资源快不够了，忍痛将某些服务先关掉，待渡过难关，再开启回来。

**所以从上述分析来看，两者其实从有些角度看是有一定的类似性的：**

目的很一致，都是从可用性可靠性着想，为防止系统的整体缓慢甚至崩溃，采用的技术手段；

最终表现类似，对于两者来说，最终让用户体验到的是某些功能暂时不可达或不可用；

粒度一般都是服务级别，当然，业界也有不少更细粒度的做法，比如做到数据持久层（允许查询，不允许增删改）；

自治性要求很高，熔断模式一般都是服务基于策略的自动触发，降级虽说可人工干预，但在微服务架构下，完全靠人显然不可能，开关预置、配置中心都是必要手段；

**而两者的区别也是明显的：**

触发原因不太一样，服务熔断一般是某个服务（下游服务）故障引起，而服务降级一般是从整体负荷考虑；

管理目标的层次不太一样，熔断其实是一个框架级的处理，每个微服务都需要（无层级之分），而降级一般需要对业务有层级之分（比如降级一般是从最外围服务开始）

实现方式不太一样；

**熔断器实现的三个状态机**

Closed：熔断器关闭状态，调用失败次数积累，到了阈值（或一定比例）则启动熔断机制；

Open：熔断器打开状态，此时对下游的调用都内部直接返回错误，不走网络，但设计了一个时钟选项，默认的时钟达到了一定时间（这个时间一般设置成平均故障处理时间，也就是MTTR），到了这个时间，进入半熔断状态；

Half-Open：半熔断状态，允许定量的服务请求，如果调用都成功（或一定比例）则认为恢复了，关闭熔断器，否则认为还没好，又回到熔断器打开状态；

Hystrix Timeouts And Ribbon Clients

Client Side Load Balancer: Ribbon

External Configuration: Archaius

Router and Filter: Zuul

Spring Cloud OpenFeign

目前，在Spring cloud 中服务之间通过restful方式调用有两种方式

- restTemplate+Ribbon

- feign

从实践上看，采用feign的方式更优雅（feign内部也使用了ribbon做负载均衡）

Spring Cloud Stream

Spring Cloud Consul

Service discovery and configuration management with Hashicorp Consul.

Spring Cloud Consul features:

* Service Discovery: instances can be registered with the Consul agent and clients can discover the instances using Spring-managed beans
* Supports Ribbon, the client side load-balancer via Spring Cloud Netflix
* Supports Zuul, a dynamic router and filter via Spring Cloud Netflix
* Distributed Configuration: using the Consul Key/Value store
* Control Bus: Distributed control events using Consul Events

Spring Cloud Zookeeper

Spring Cloud Bus

Spring Cloud Sleuth





