Spring cloud 微服务框架简介

针对当前流行的微服务架构，spring cloud 提供一整套解决方案，通过构建其框架下的各个组件可快速实现微服务设计中的相关功能：

- 服务注册与发现
- 服务网关
- 服务通信
- 服务治理
- 配置管理

等等

Spring cloud 针对各个功能都有对应的组件框架可供选择使用。

Spring-cloud-Netflix

Spring Cloud Netflix 提供了对 Netflix 开源项目的集成，该项目是 Spring Cloud 的子项目之一，使得我们可以以 Spring Boot 编程风格使用 Netflix 旗下相关框架。只需要在程序中添加注解，就能使用成熟的 Netflix 组件来快速实现分布式系统的常见架构模式。这些模式包括服务发现(Eureka),智能路由(Zuul)和客户端负载均衡(Ribbon), 断路器(Hystrix)。

Eureka

Eureka 是 Spring Cloud Netflix 的子项目，提供在分布式环境下的服务发现，服务注册的功能。

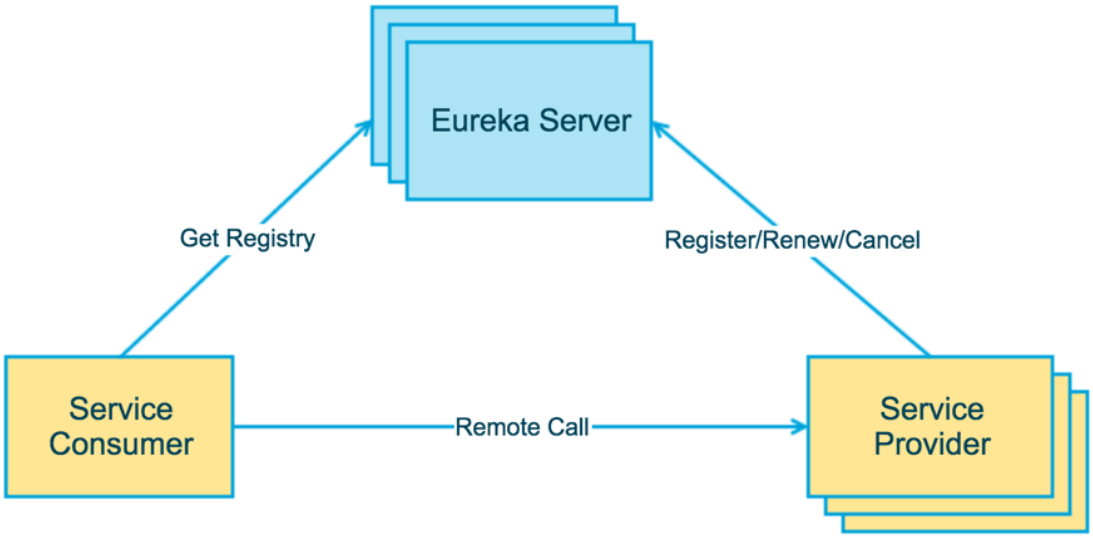
作用

- **服务注册**
  客户端向 Eureka 注册并提供一些元信息，如主机名、端口号、获取健康信息的 url 和主页等。Eureka 通过心跳连接判断服务是否在线，如果心跳检测失败超过指定时间，对应的服务通常就会被移出可用服务列表
- **服务发现**
  Eureka 把所有注册信息都放在内存中，所有注册过的客户端都会向 Eureka 发送心跳包来保持连接。客户端会有一份本地注册信息的缓存，这样就不需要每次远程调用时都向 Eureka 查询注册信息。
  默认情况下，Eureka 服务端自身也是个客户端，所以需要指定一个 Eureka Server 的 URL 作为"伙伴"(peer)。如果你没有提供这个地址，Eureka Server 也能正常启动工作，但是在日志中会有大量关于找不到 peer 的错误信息。

原理

组件构成：

- **Eureka-server:**
  服务注册中心，存储所有的注册服务信息，根据客户端上报的心跳检查，定期清理无效服务
- **Eureka-client:**
  一个 java 客户端，内嵌入业务服务模块，用来简化与服务器的交互，启动的时候，会初始化多个定时任务：
  1) 定时的把本地的服务配置信息，即需要注册到远端的服务信息自动刷新到注册服务器上
  2) 定时的获取远端的注册信息
  3) 定时上报本地服务健康状况（心跳检查）
  4) 作为轮询负载均衡器，并提供服务的故障切换支持



特点

- **高可用性：**
  Eureka 对 CAP 理论中可保证 AP，而牺牲一致性，针对 service 发现这种非强一致的场景，虽然会有部分情况下没发现新服务导致请求出错，但不会因为数据的不同步问题导致全集群不可用
- **自我保护模式：**
  Eureka 针对网络分区问题，提供一种“自我保护”模式策略，即当 Eureka 服务节点在短时间里丢失了大量的心跳连接（注：可能发生了网络故障），此时 Eureka 节点会进入”自我保护模式“，同时保留那些“心跳死亡“的服务注册信息不过期。此时，这个 Eureka 节点对于新的服务还能提供注册服务，对于”死亡“的仍然保留，以防还有客户端向其发

起请求。当网络故障恢复后，这个 Eureka 节点会退出”自我保护模式“。

- **基于 HTTP：**
  Eureka 是针对 REST 服务，提供的注册发现中心，基于 Http 的通信机制的中间层服务，有对云环境天生的部署支持，可很方便的接入公有云等 Paas 平台

Zuul

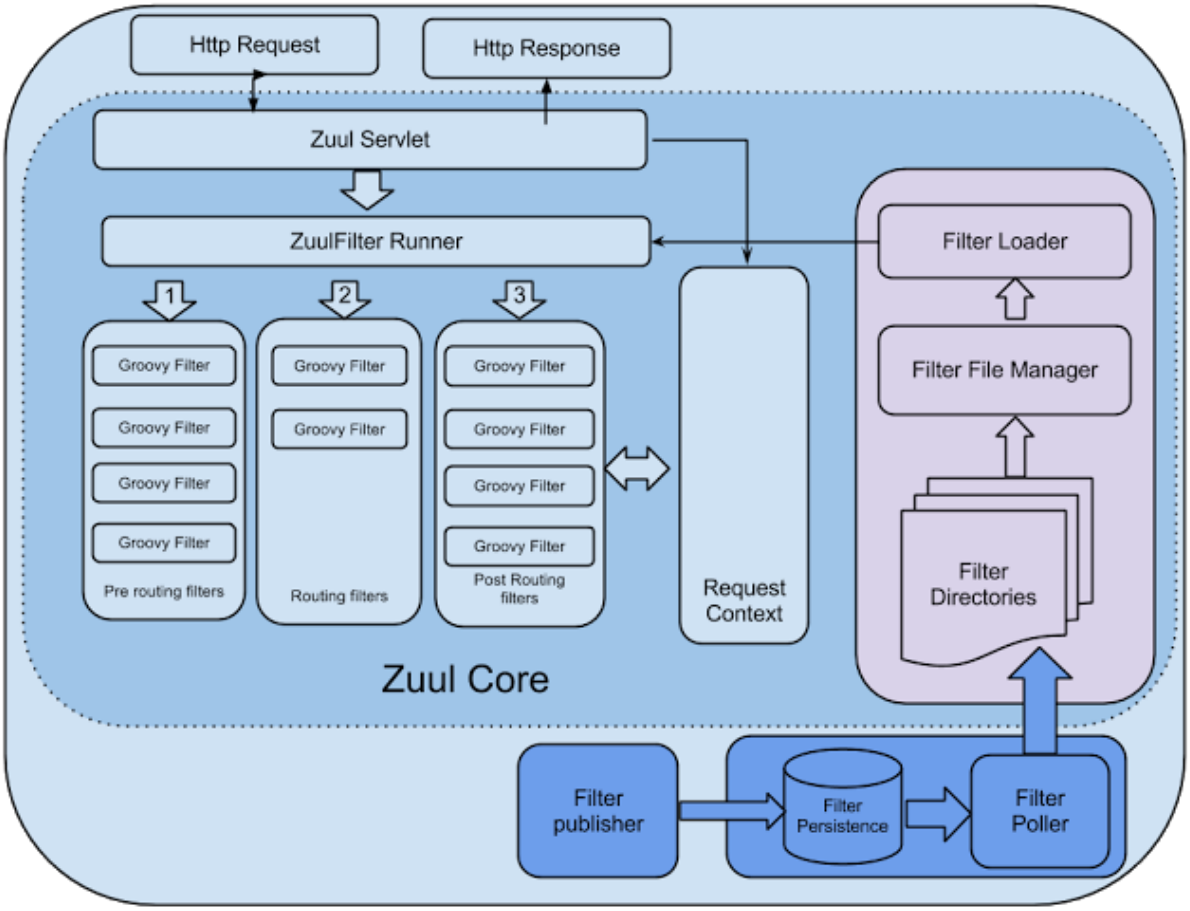
Zuul 是 Spring Cloud Netflix 的子项目，提供在分布式环境下智能路由、反向代理等网关功能

作用

- **智能路由：**
  以动态方式根据需要将请求路由至不同后端集群处理
- **安全与验证：**
  识别面向不同资源的验证要求并拒绝那些与要求不符的请求
- **静态响应处理：**
  在请求入口位置直接建立部分响应，从而避免静态资源访问流入内部动态服务集群
- **流量整形：**
  为不同负载类型分配对应容量，并弃用超出限定值的请求
- **多区域弹性:**
  跨越 AWS 区域进行请求路由，旨在实现 ELB 使用多样化并保证网关位置与使用者尽可能接近

原理

Zuul 通过 filter 机制对请求进行拦截，并根据相关配置策略进行后期处理。
Zuul 框架实际是对过滤器进行动态的加载，编译，运行。过滤器之间没有直接的相互通信，而是通过一个 RequestContext 的静态类来进行数据传递的。RequestContext 类中有 ThreadLocal 变量来记录每个 Request 所需要传递的数据。



Ribbon

Ribbon 是 Spring Cloud Netflix 的子项目，提供在客户端的负载均衡算法，将 Netflix 的中间层服务连接在一起。

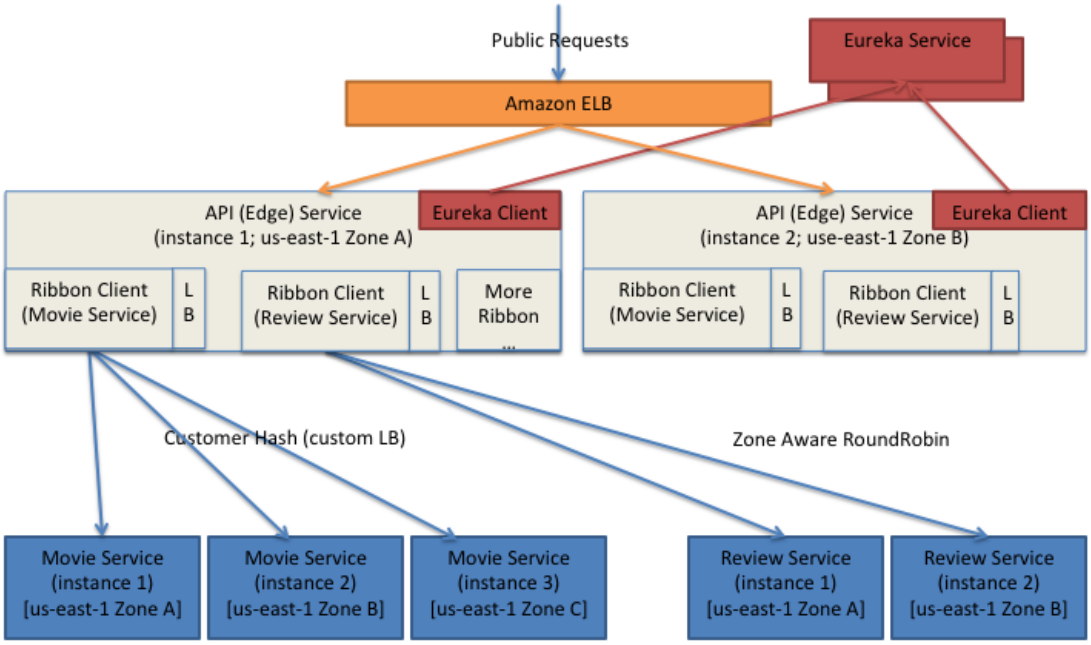
作用

- **负载均衡：**
  Ribbon 内置可插拔、可定制的负载均衡组件并提供如下负载均衡策略：
  1) 简单轮询负载均衡
  2) 加权响应时间负载均衡
  3) 随机负载均衡
  4) 区域感知轮询负载均衡（针对 AWS）
- **Failover：**
  Ribbon 客户端组件提供一系列完善的配置项以实现：连接超时，重试等
- **集成 Eureka:**
  可与 Eureka 无缝集成，实现在动态环境下的服务发现与负载均衡

原理

Ribbon 的两种运行方式：

1) **配置 Ribbon Server List:**
   通过在客户端中配置的 ribbon Server List 服务端列表去轮询访问以达到均衡负载的作用
2) **Ribbon 与 Eureka 联合使用：**
   Ribbon 集成 Eureka 后，ribbonServerList 会被 DiscoveryEnabledNIWSServerList 重写，扩展成从 Eureka 注册中心中获取服务端列表，各客户端通过 Eureka 注册中心自动发现已注册的服务列表以实现负载均衡



Hystrix

复杂的分布式架构应用程序都存在大量的依赖调用，在高并发的依赖调用失败时，需要相应的隔离措施与降级策略，以防依赖调用不可用导致一系列请求调用阻塞而发生系统雪崩。

Hystrix 是 Spring Cloud Netflix 的子项目，提供了一种断路器(Cricuit Breaker)模式，实现在远程服务不可用时自动熔断(打开开关)，并在远程服务恢复时自动恢复(闭合开关)。

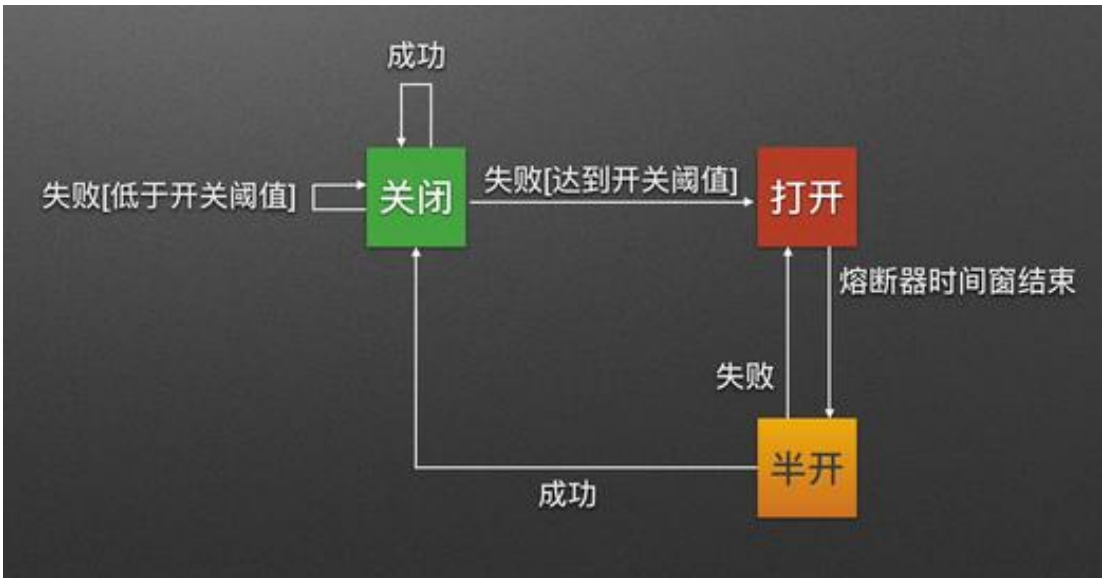
Hystrix 实质是一套服务治理与降级的框架，针对分布式系统下依赖调用的延迟和容错库，提供延迟和容错功能，隔离远程系统、和访问第三方程序库的访问点，防止级联失败，保证复杂的分布系统在面临不可避免的失败时，仍能有其弹性。

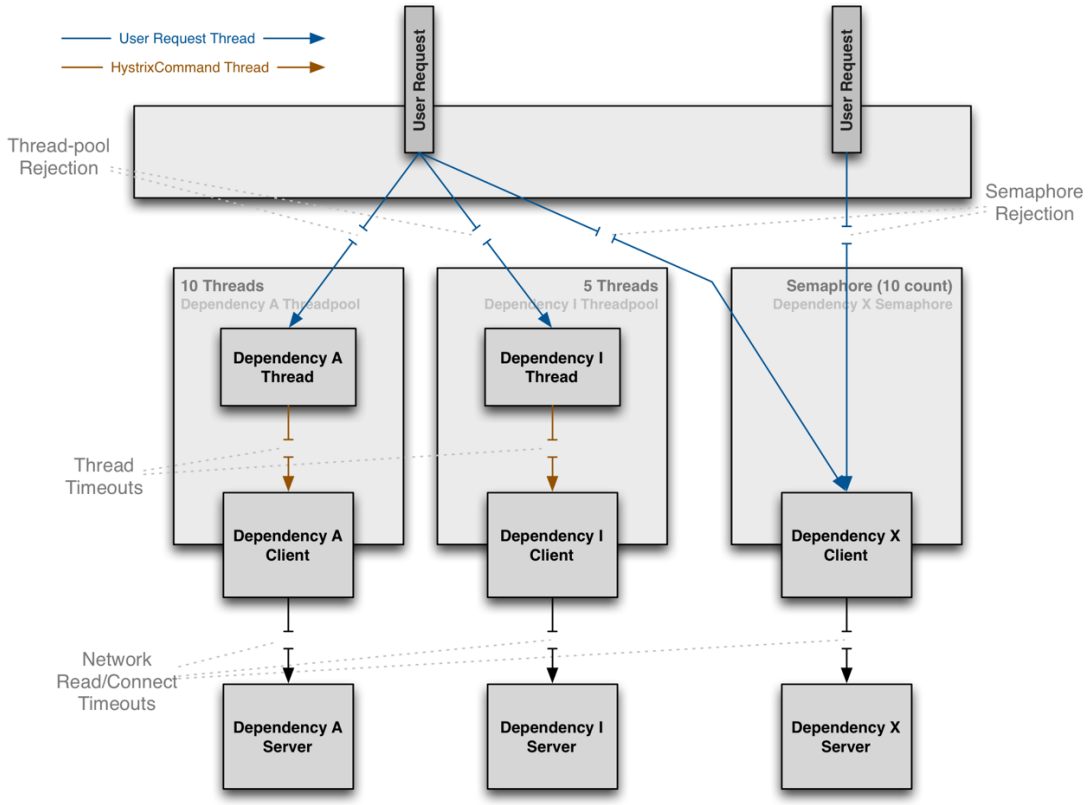
作用

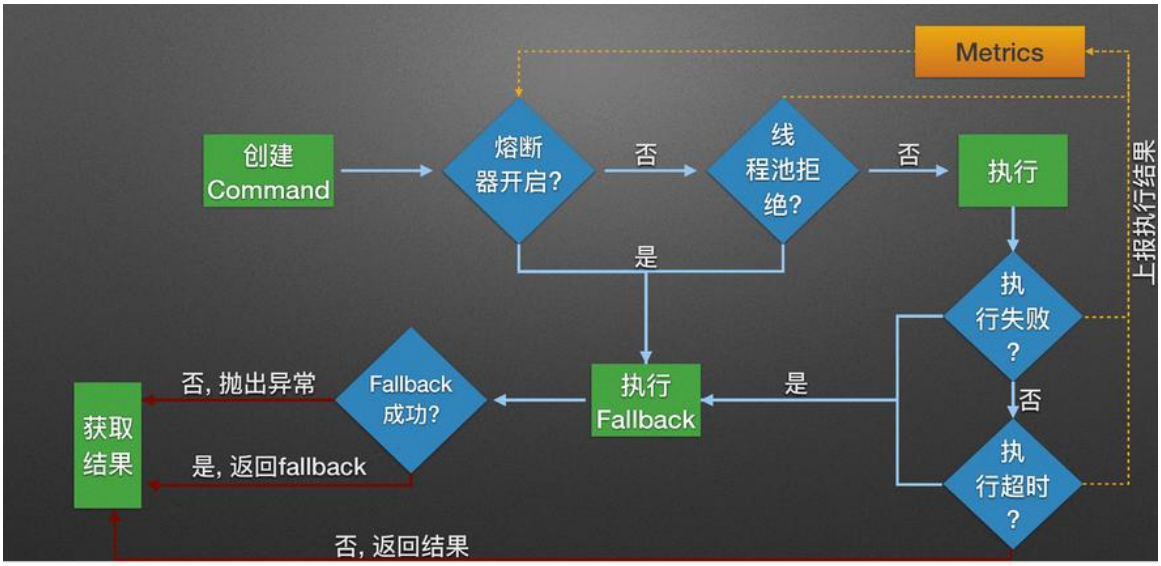
- **熔断器：**
  Hystrix 通过配置请求响应的错误率阈值，定义一个熔断开关，当达到阈值时，可以自动运行或手动调用,停止当前依赖一段时间(10 秒)，熔断器默认错误率阈值为 50%,超过将自动运行
- **超时控制：**
  可配置依赖调用超时时间,超时时间一般设为比 99.5%平均时间略高即可.当调用超时时，直接返回或执行 fallback 逻辑
- **资源隔离：**
  采用线程/信号的方式,通过隔离限制依赖的并发量和阻塞扩散，为每个依赖提供一个独立的小的线程池（或信号），如果线程池已满调用将被立即拒绝，默认不采用排队.加速失败判定时间
- **Fallback 降级：**
  依赖调用结果分为:成功，失败（抛出异常），超时，线程拒绝，短路。 请求失败(异常，拒绝，超时，短路)时执行 fallback(降级)逻辑

原理

Hystrix 使用命令模式 HystrixCommand(Command)包装依赖调用逻辑，每个命令在单独线程中/信号授权下执行，当 Hystrix 执行命令超时后，，会尝试去调用一个 fallback 方法，这个 fallback 即一个备用方案的降级策略，要为 HystrixCommand 提供 fallback。
同时 Hystrix 会统计命令调用，看其中失败的比例，默认当前系统响应的健康状况超过一定阈值的失败后，开启熔断器，之后一段时间窗口内的命令调用直接返回失败（或者走 fallback），时间窗口期结束后，熔断器会自动进入半开状态，这时熔断器只允许一个请求通过，当该请求调用成功时，Hystrix 再尝试关闭熔断器，看看请求是否能正常响应。




User Request Thread
HystrixCommand Thread
User Request
User Request
Thread-pool Rejection
Semaphore Rejection
10 Threads
Dependency A Threadpool
5 Threads
Dependency I Threadpool
Semaphore (10 count)
Dependency X Semaphore
Dependency A Thread
Dependency I Thread
Thread Timeouts
Dependency A Client
Dependency I Client
Dependency X Client
Network Read/Connect Timeouts
Dependency A Server
Dependency I Server
Dependency X Server


Metrics
创建 Command
熔断器开启?
否
线程池拒绝?
否
执行
是
执行失败?
上报执行结果
否
执行超时?
执行 Fallback
是
Fallback 成功?
否, 抛出异常
获取结果
是, 返回fallback
否, 返回结果