1. Springcloud

服务注册:

spring-web容器启动后,会发送servelt容器初始化完成事件,

服务注册组件AbstractAutoServiceRegistration会监听改事件然后,

调用ServiceRegistry接口的register()(不同的注册中心实现不一样),向注册中心注册的服务配置信息,其中包括服务serviceid与ip+端口.

通常会以Rest接口的调用的形式将服务信息,注册到注册中心上.

服务发现:

SpringCloud服务发现是伴随SpringCloud-Ribbon的ILoadBalancer; ILoadBalancer的默认实现是DynamicServerListLoadBalancer(动态负载均衡),

内部维护服务列表, 服务列表是通过 ServerListUpdater接口进行更新, 也就是服务发现是通过ServerListUpdater实现的, ServerListUpdater默认是一个

定时任务,根据我们要发现服务的serviceid,到注册中心获取服务信息列表(Rest接口),然后与本地缓存的服务列表比对,如果本地不存在则增加到本地缓存,

如果远程不存在则剔除本地的服务.

服务调用:

伴随着客户端的负载均衡策略 。默认会使用 roundribbon randomrule RetryRule

负载均衡:

服务发现时我们获取了服务注册信息的列表(services),我只可以指定负载均衡策略(IRule接口实现): 获取对应的service,

services中封装着着service的ip+端口信息,然后我们同ip+端口调用服务

限流熔断:

1.hystrix

hystrix是通过RX(一个响应是框架,基于JAVA观察者模式),与滑动窗口实现的,是纯粹的线程池资源绝对隔离. 首先被hystrix保护的资源,会被封装成一个hystrixCommand其中包括着

hystrix CommandKey与ThreadPoolKey, 通过RX的响应式框架将hystrixCommand 封装成一个Observable(被观察者),

并为其添加了一系列的观察者,观察将上报hystrixCommandKey的执行情况到Hystrix的统计组件,统计使用滑动窗口统计(RX timewindows),将异常比例

达到阈值是,断路器将打开.所有的hystrixCommand将全部fallBack.

最后通过ThreadPoolKey获取ThreadPool 将hystrixCommand封装成一个RX的异步Feature. 于此同时注意线程同步获取Feature的结果,并将结果返回

2.sentinel

hystrix的劣势: ThreadPoolKey使用不当讲导致大量创建线程池; 线程池隔离导致线程资源消耗,线程上下文切换消耗. hystrix现在已经不在维护

sentinel优势: 轻量级; 使用内存数组进行数据统计性能优于hystrix滑动窗口(RX timewindows), 支持动态配置调整,支持限流, 强大的 dashboard功能

Gateway:

为什么需要Gateway: 开发公网入口给外部服务掉用 ---> 公网调用鉴权(appKey + auth系统鉴权), 验签(MD5摘要--唯品会那种)

路由转发: 通过Host路由 ---> xxx-api.abc.com ----> xxx.api.abc.com ---> 内部NG负载均衡 ----> 目标服务