登录 注册 博客 程序员学院 下载 论坛 问答 代码 直播 硬件小屋 商城 java实时获取最全资讯 搜索 会员中心 收藏 动态 消息 创作中心

SpringCloud 熔断分数配置说明

原创 微博小屋 2018-11-13 17:26:42 5561 收藏 3
分类专栏： SpringCloud

Spring Cloud Hystrix（服务容错保护）

2017年06月26日 17:52:42 jack2017106 阅读数：3763

Hystrix介绍

简介

Hystrix是由Netflix创建一个组件。

在微服务的分布式环境中，系统内存在许多服务依赖，在高并发的环境下，这些依赖的稳定性与否对系统的影响非常大，但是依赖有很多不可控问题：如网络连接缓慢，资源耗尽，暂时不可用，服务超时等，Hystrix可以通过超时设置和熔断机制来保证部分有限度的不影响整个系统的稳定性。Hystrix通过隔离服务之间的接入点，阻止它们之间的级联故障，并提供备用选项，从而提高系统的整体弹性。

Hystrix流程结构

流程说明:

1. 每次调用创建一个新的HystrixCommand，把依赖调用封装在run()方法中
2. 执行execute()/queue()做同步或异步调用
3. 查看开启缓存没有？如果开启了，并且缓存操作时间窗内的缓存可用，则此请求直接返回这个Observable'的形式返回。
4. 判断熔断器(circuit-breaker)是否打开，如果打开就跳到步骤8，进行降级策略，如果打开没有，则继续
5. 判断线程池/队列/信号量是否跑满，如果跑满就进入降级步骤8，否则继续后续步骤
6. 调用HystrixCommand的run()方法。运行依赖逻辑
   1. 依赖逻辑调用超时，进入步骤8
7. 判断逻辑调用响应的
   1. 返回成功调用结果
   2. 调用出错，进入步骤8
8. 计算熔断器状态，所有的运行状态(成功, 失败, 拒绝, 超时)上报给熔断器，用于统计从而判断熔断器状态
9. getFallback()降级逻辑。
   1. 以下四种情况将触发getFallback()调用：
      1. (1) run()抛出非HystrixBadRequestException异常。
      2. (2) run()方法调用超时
      3. (3) 熔断器开启打断调用
      4. (4) 线程池/队列/信号量是否跑满
   2. 没有实现getFallback()的Command将直接抛出异常
   2. fallback降级逻辑调用成功直接返回
   3. 降级逻辑调用失败抛出异常
10. 返回执行成功结果

为什么要使用Hystrix

分布式微服务系统中有许多依赖，在高并发的环境下这些依赖的稳定性与否对系统的影响非常大，但是依赖有很多不可控问题：如网络连接缓慢，资源耗尽，暂时不可用，服务超时等，高并发情况下，某个依赖服务出现下面的Vehicle Service(车况，其他依赖服务可用。

如果不对大规模依赖采取隔离措施，将会生产雪崩效应，最终可能导致当前应用服务器有被拖垮的风险。

解决以上问题的方案-断路器：

断路器（英文名称：circuit-breaker，circuit breaker）是微服务中，承载和开启某项目服务并下的电流并超负荷，在现实世界的电力承载和开启保护设置的断路器并下的电流的开关装置。

下图是实现的场景

Hystrix实现的断路器模式，在分布式微服务系统中当对特定服务的呼叫达到一定阈值时（Hystrix中默认为20秒，20秒），电路将打开，不进行通讯，在错误时间窗的情况下，开发人员可以提供回退

还是回到呢？的说法回退都是有条件的，在有条件是一般是不得同步都是有关联，达到新连接时自的阈值，断路器都打开，对应 的Fallback方法被默认，一般的说的是于会有所关系的服务发起请求。

资料

官方文档

http://cloud.spring.io/spring-cloud-static/Dalston.SR1/#_circuit_breaker_hystrix_clients

源码地址

https://github.com/Netflix/Hystrix

博文推荐

- Hystrix 使用与分析 http://hot6hot.iteye.com/blog/2155636

如何集成Hystrix

step1. 在消费者微服务的pom.xml中添加Hystrix依赖

展开源码

step2. 在启动程序中启用断路器

展开源码

step3. 在具体方法中添加注解并添加FallBack方法

展开源码

Hystrix配置

配置样例:

```
@HystrixCommand (fallbackMethod = "stubMyServiceFallback" ,
                 commandProperties = {
        @HystrixProperty (name= "execution.isolation.strategy" , value= "SEMAPHORE" )
                 }
)
```

Execution相关的属性的配置：

- hystrix.command.default.execution.isolation.strategy 隔离策略，默认是Thread, 可选Thread | Semaphore
  - thread 通过线程数量来限制并发请求数，可以提供超时中的保护，但有一定的延迟。一般用于网络调用
  - semaphore 通过semaphore count来限制并发请求数，适用于无网络延离并发请求
- hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds 命令执行超时时间，默认1000ms
- hystrix.command.default.execution.timeout.enabled 执行是否启用超时，默认启用true
- hystrix.command.default.execution.isolation.thread.interruptOnTimeout 发生超时是是否中断，默认true
- hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests 最大并发请求数，默认10，该参数会使用ExecutionIsolationStrategy.SEMAPHORE策略时才有效。如果达到最大并发请求数，请求会被拒绝。理论上选择semaphore size的原则和选择thread size一样，但选择semaphore时每次执行的单元要比较小且执行速度快（ms级别），否则的话应该采用thread。semaphore应该占据了容器（tomcat）的线程池的一个部分。

Fallback相关的属性

这些参数可以应用于Hystrix的THREAD和SEMAPHORE策略

- hystrix.command.default.fallback.isolation.semaphore.maxConcurrentRequests 如果并发数达到该设置值，请求会被拒绝和抛出异常并且fallback不会被调用，默认10
- hystrix.command.default.fallback.enabled 当执行失败或者请求被拒绝，是否会尝试调用hystrixCommand.getFallback()。默认true

Circuit Breaker相关的属性

- hystrix.command.default.circuitBreaker.enabled 用来跟踪circuit的健康性，如果未达到阈以request阈值。默认true
- hystrix.command.default.circuitBreaker.requestVolumeThreshold 一个rolling window内最小的调用数。如果设为20，那么当一个rolling window的时间内（比如说1个rolling window是10秒）收到19个请求，即使19个请求都失败，也不会触发circuit break。默认20
- hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds 触发短路的时间值，当该值设为5000时，则当触发circuit break后的5000毫秒内都会拒绝request，但是5000秒钟过后才会关闭circuit。默认5000
- hystrix.command.default.circuitBreaker.errorThresholdPercentage错误比率阈值，如果错误率>=该值，circuit会被打开，并拒绝所有请求触发fallback。默认50
- hystrix.command.default.circuitBreaker.forceOpen 强制打开熔断器，如果打开这个开关，那么拒绝所有request，默认false
- hystrix.command.default.circuitBreaker.forceClosed 强制关闭熔断器 如果这个开关打开，circuit将一直关闭且忽略circuitBreaker.errorThresholdPercentage

Metrics相关参数

- hystrix.command.default.metrics.rollingStats.timeInMilliseconds 设置统计时间窗口值的，毫秒值，circuit break 的打开会根据1个rolling window的统计来计算。若rolling window被设为10000毫秒，则rolling window会被分成n个buckets，每个bucket包含success，failure，timeout，rejection的次数的统计信息。默认10000
- hystrix.command.default.metrics.rollingStats.numBuckets 设置一个rolling window被划分的数量，若numBuckets = 10，rolling window = 10000，那么一个bucket的时间窗口是1s，必须符合rolling window % numberBuckets == 0。默认10
- hystrix.command.default.metrics.rollingPercentile.enabled 执行时是否enable指标的计算跟踪，默认true
- hystrix.command.default.metrics.rollingPercentile.timeInMilliseconds 设置rolling percentile window的时间，默认60000
- hystrix.command.default.metrics.rollingPercentile.numBuckets 设置rolling percentile window的numberBuckets。逻辑同上。默认6
- hystrix.command.default.metrics.rollingPercentile.bucketSize 设置bucket size = 100，window = 10s，若这10s里有5000次执行，只有最后100次执行会被统计到bucket里去。增加该值会增加内存开销以及排序的开销。默认100
- hystrix.command.default.metrics.healthSnapshot.intervalInMilliseconds 记录health快照（用来统计成功和错误率）的间隔，默认500ms

Request Context相关参数

hystrix.command.default.requestCache.enabled 默认true，需要重载getCacheKey()，返回null时不缓存
hystrix.command.default.requestLog.enabled 记录日志到HystrixRequestLog 默认true

Collapser Properties相关参数

hystrix.collapser.default.maxRequestsInBatch 单次批处理的最大请求数，达到该数量触发批处理，默认Integer.MAX_VALUE
hystrix.collapser.default.timerDelayInMilliseconds 触发批处理的延迟，也可以为创建批处理的时间 + 该值，默认10
hystrix.collapser.default.requestCache.enabled 是否对HystrixCollapser.execute() and HystrixCollapser.queue()的cache，默认true

ThreadPool相关参数

线程数默认值10适用于大部分情况（有时可以设置得更小），如果需要设置得更大，那有个基本得公式可以follow:
requests per second at peak when healthy × 99th percentile latency in seconds + some breathing room
每秒最大支撑的请求数 (99% 的平均响应时间) + 缓冲器
比如：每秒能够扛900个请求，99%的请求的响应时间是60ms，那么公式是：
900 * (0.060+0.012)

基本得原则时保持线程池尽可能小，性能衰退为了释放压力，防止资源被耗尽。
当一个线程池正常时间时，线程池一般只会有1到2个线程处理其他请求的请求

- hystrix.threadpool.default.coreSize 开发执行回调之线程数，默认10
- hystrix.threadpool.default.maxQueueSize BlockingQueue的最大队列长度，当设为 -1，会使用SynchronousQueue，值为正时使用LinkedBlockingQueue。该设置只会在配置项allowMaximumSizeToDivergeFromCoreSize被启用的情况下才有意义，默认是-1。
- hystrix.threadpool.default.queueSizeRejectionThreshold 即便maxQueueSize没有达到，达到queueSizeRejectionThreshold设置后，请求会被拒绝。因为maxQueueSize不能被动态修改，这个参数将允许我们对队列的设置进行调整。if maxQueueSize == -1，该字段将不起作用
- hystrix.threadpool.default.keepAliveTimeMinutes 设置corePoolSize和maxPoolSize动态一样（默认设置）该设置无效。如果自定义plugin（https://github.com/Netflix/Hystrix/wiki/Plugins）使用自定义实现，该设置才有用，默认1
- hystrix.threadpool.default.metrics.rollingStats.timeInMilliseconds 线程池统计指标的时间，默认10000
- hystrix.threadpool.default.metrics.rollingStats.numBuckets 滚动窗 rolling window的分段buckets，默认10

官方说明：

https://github.com/Netflix/Hystrix/wiki/Configuration

HystrixCommandProperties默认配置 White_Lee的专栏 ⓘ 392
下面是HystrixCommandProperties的默认配置参数。 HystrixCommandProperties.Setter commandProperties = HystrixCommandProperties.Setter()...
SpringCloud两种服务降级的方法 不调皮的 ⓘ 4169
1.fallbackfallbackfallback2.Hystrix熔断器3.注解fallbackfallbackfallbackfallbackfallbackfallbackfallbackfallback.pojo.impl.controller.fallbackfallbackfallbackfallbackfallbackfallbackfallbackf