Hystrix落地方案书

一、需求背景

API后台接口或者WAP现在基本都是依赖其他组的SDK或者EJB，当其他组服务不可用的时候，很有可能引发雪崩效应，导致整个系统不可用，这个时候就需要服务降级和熔断。

|  |
| --- |
|  |

二、Hystrix介绍

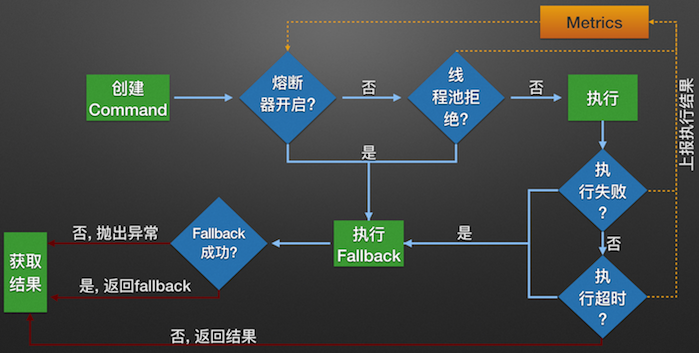
### 简介

Hystrix源于Netflix API团队在去年启动的弹性工程项目，在此期间，Hystrix得到了不断发展，并逐渐成熟。现在，在Netflix网站中，每天有数十亿的独立线程和信号通过Hystrix进行调用，Hystrix的运行时间和弹性也得到了显著的改善。

### 作用

* 控制被依赖服务的延时和失败
* 防止在复杂系统中的级联失败
* 可以进行快速失败（不需要等待）和快速恢复（当依赖服务失效后又恢复正常，其对应的线程池会被清理干净，即剩下的都是未使用的线程，相对于整个 Tomcat 容器的线程池被占满需要耗费更长时间以恢复可用来说，此时系统可以快速恢复）
* getFallback（失败时指定的操作）和优雅降级
* 实现近实时的检测、报警、运维

### 流程



三、Hystrix配置

### 核心配置

* 隔离模式

hystrix默认的是线程池模式，所有请求都会入队列处理，信号量模式就是当并发量达到系统设置的临界点时候就会丢弃这些请求直接进入到fallback。

* 熔断比例

当我们已知某个服务不可用的时候可以使用hystrix优先级最高配置进行强制熔断，这样接下里就不会进行重试，当然还有一些其他的配置如下图。

* 线程池大小

每一个服务我们可以抽象出来为一个command，我们需要对调用频率高的服务设置线程池大点。

* 超时时间

超时时间默认是1s，这个参数开始时候我们可以设置大一点，然后通过调用日志，再灵活配置。

|  |
| --- |
|  |

四、落地方案

### cs后台配置

新增hystrixcommand表，用来储存每一个command的配置信息



### 代码层实现

为了无缝接入现有的代码，hystrix将以aop形式去代理所有需要拦截的接口，当然特殊的接口可能需要编写定制的容错返回数据。

所有接入项目都会定时的去同步hystrix配置信息

五、hystrix接入业务场景示例

### 秒杀VI页面

秒杀VI页的数据除了库存量会发生变动外，其他信息基本不会变动，虽然现在底层接口数据是放redis，每次查询都是走缓存，但是不敢保证基础服务的的redis高可用，一旦底层挂了，我们的VI页就会显示活动不存在，然后就各种投诉到我们这边了。

为了防止这种情况，我们需要接入hystrix，当我们正常调用接口时候可以把正确的数据放到ehcache里面，每次正常调用都会按照一定的规则去刷新ehcache里面的值。当VI页详情接口挂了时候，我们直接返回ehcache里面容错的数据。这样至少能让用户看到，下单的时候就算失败了也会提示活动太火爆，对用户来说是无感知的。