27-API网关：系统的门面要如何做呢？

你好，我是唐扬。

到目前为止，你的垂直电商系统在经过微服务化拆分之后，已经运行了一段时间了，系统的扩展性得到了很大的提升，也能够比较平稳地度过高峰期的流量了。

不过最近你发现，随着自己的电商网站知名度越来越高，系统迎来了一些“不速之客”，在凌晨的时候，系统中的搜索商品和用户接口的调用量，会有激剧的上升，持续一段时间之后又回归正常。

这些搜索请求有一个共同特征是，来自固定的几台设备。当你在搜索服务上加一个针对设备ID的限流功能之后，凌晨的高峰搜索请求不见了。但是不久之后，用户服务也出现了大量爬取用户信息的请求，商品接口出现了大量爬取商品信息的请求。你不得不在这两个服务上也增加一样的限流策略。

但是这样会有一个问题：不同的三个服务上使用同一种策略，在代码上会有冗余，无法做到重用，如果其他服务上也出现类似的问题，还要通过拷贝代码来实现，肯定是不行的。

不过作为Java程序员，**你很容易想到：**将限流的功能独立成一个单独的jar包，给这三个服务来引用。不过你忽略了一种情况，那就是你的电商团队使用的除了Java，还有PHP和Golang等多种语言。

用多种语言开发的服务是没有办法使用jar包，来实现限流功能的，**这时你需要引入API网关。**

API网关起到的作用（904）

API网关（API Gateway）不是一个开源组件，而是一种架构模式，它是将一些服务共有的功能整合在一起，独立部署为单独的一层，用来解决一些服务治理的问题。你可以把它看作系统的边界，它可以对出入系统的流量做统一的管控。

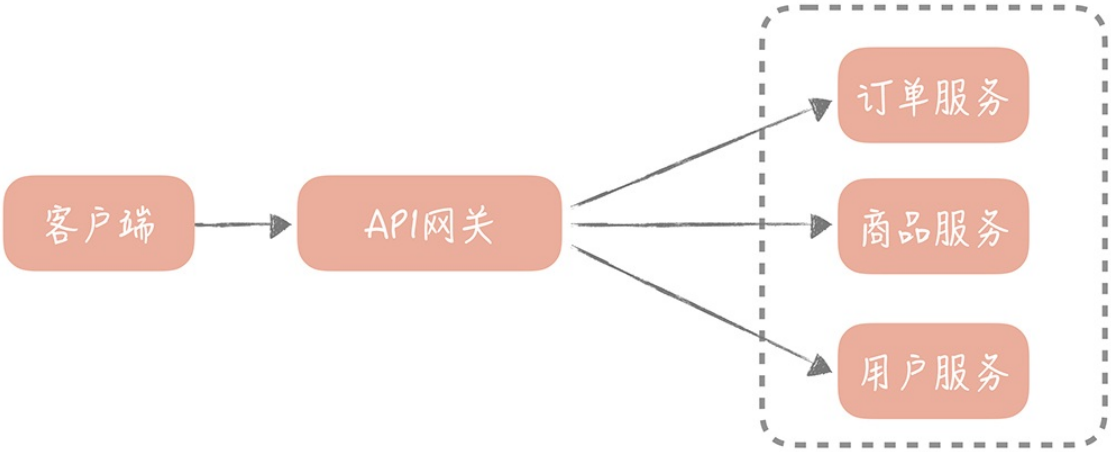
在我看来，API网关可以分为两类：**一类叫做入口网关，一类叫做出口网关。**

入口网关是我们经常使用的网关种类，它部署在负载均衡服务器和应用服务器之间，**主要有几方面的作用。**

1. 它提供客户端一个统一的接入地址，API网关可以将用户的请求动态路由到不同的业务服务上，并且做一些必要的协议转换工作。**在你的系统中，你部署的微服务对外暴露的协议可能不同：**有些提供的是HTTP服务；有些已经完成RPC改造，对外暴露RPC服务；有些遗留系统可能还暴露的是Web Service服务。API网关可以对客户端屏蔽这些服务的部署地址，以及协议的细节，给客户端的调用带来很大的便捷。
2. 另一方面，在API网关中，我们可以植入一些服务治理的策略，比如服务的熔断、降级，流量控制和分流等等（关于服务降级和流量控制的细节，我会在后面的课程中具体讲解，在这里，你只要知道它们可以在API网关中实现就可以了）。
3. 再有，客户端的认证和授权的实现，也可以放在API网关中。你要知道，不同类型的客户端使用的认证方式是不同的。**在我之前项目中，**手机APP使用Oauth协议认证，HTML5端和Web端使用Cookie认证，内部服务使用自研的Token认证方式。这些认证方式在API网关上，可以得到统一处理，应用服务不需要了解认证的细节。
4. 另外，API网关还可以做一些与黑白名单相关的事情，比如针对设备ID、用户IP、用户ID等维度的黑白名单。

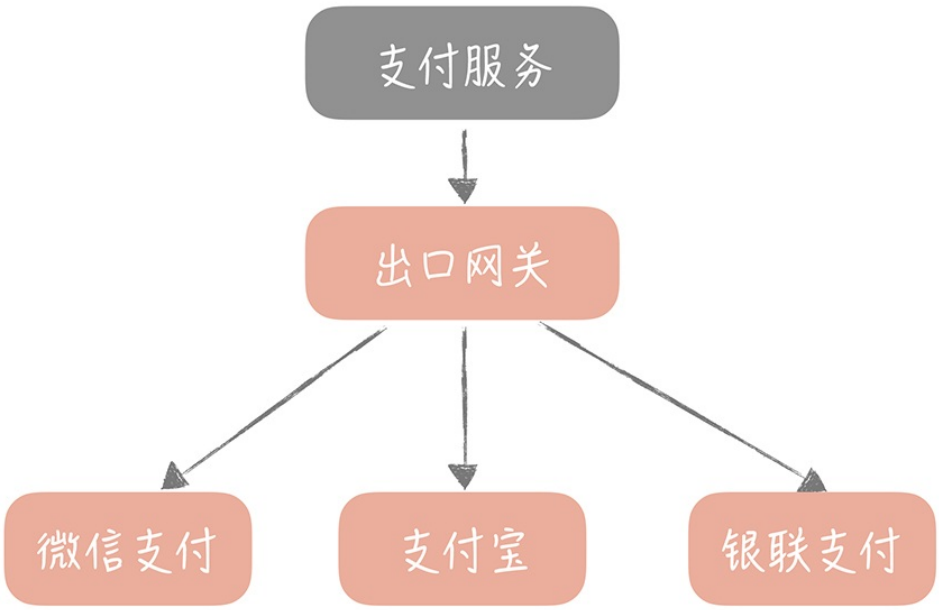
5.最后，在API网关中也可以做一些日志记录的事情，比如记录HTTP请求的访问日志，我在25讲中讲述分布

式追踪系统时，提到的标记一次请求的requestId，也可以在网关中来生成。



入口网关示意图

出口网关就没有这么丰富的功能和作用了。我们在系统开发中，会依赖很多外部的第三方系统，比如典型的例子：第三方账户登录、使用第三方工具支付等等。我们可以在应用服务器和第三方系统之间，部署出口网关，在出口网关中，对调用外部的API做统一的认证、授权，审计以及访问控制。



出口网关示意图

我花一定的篇幅去讲API网关起到的作用，主要是想让你了解，API网关可以解决什么样的实际问题，这样，当你在面对这些问题时，你就会有解决的思路，不会手足无措了。

API网关要如何实现

了解API网关的作用之后，所以接下来，我们来看看API网关在实现中需要关注哪些点，以及常见的开源API网关有哪些，这样，你在实际工作中，无论是考虑自研API网关还是使用开源的实现都会比较自如了。

在实现一个API网关时，你首先要考虑的是它的性能。这很好理解，API入口网关承担从客户端的所有流量。假如业务服务处理时间是10ms，而API网关的耗时在1ms，那么相当于每个接口的响应时间都要增加10%，这对于性能的影响无疑是巨大的。而提升API网关性能的关键还是在I/O模型上（我在23讲中详细讲到过），

这里只是举一个例子来说明I/O模型对于性能的影响。

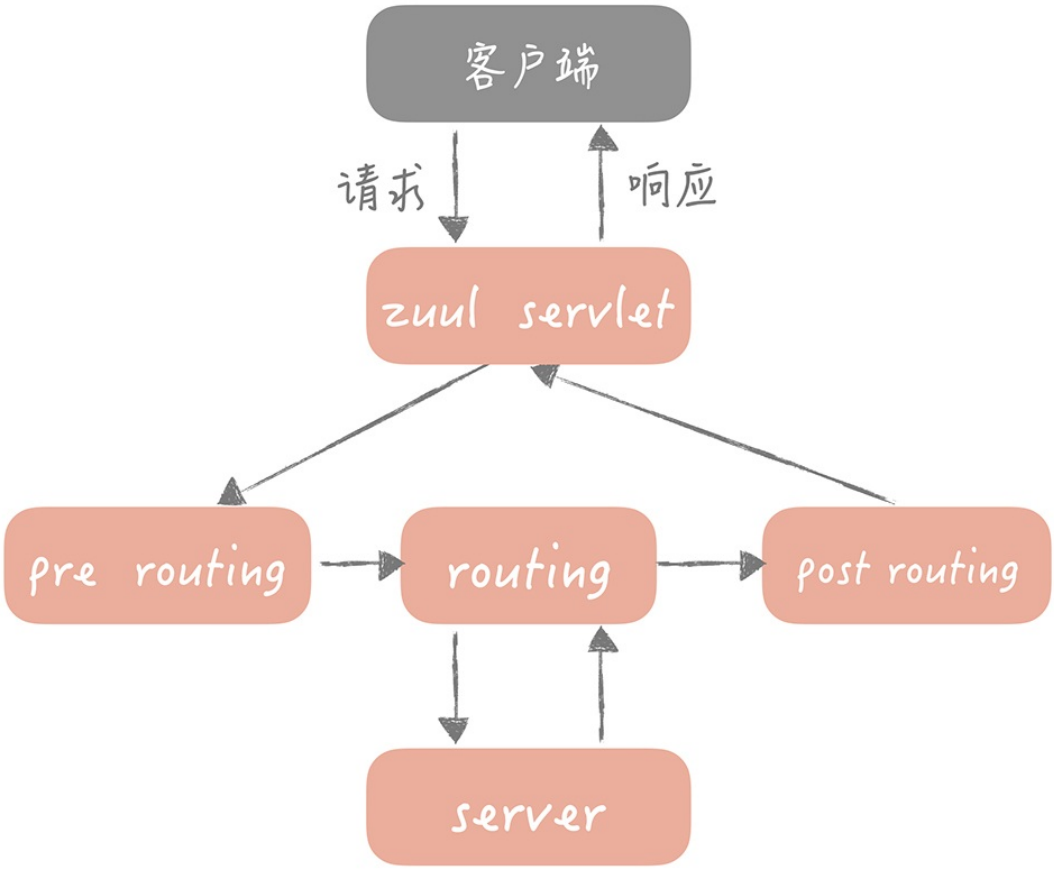
Netfix开源的API网关Zuul，在1.0版本的时候使用的是同步阻塞I/O模型，整体系统其实就是一个servlet，在接收到用户的请求，然后执行在网关中配置的认证、协议转换等逻辑之后，调用后端的服务获取数据返回给用户。

而在Zuul2.0中，Netfix团队将servlet改造成了一个netty server（netty服务），采用I/O多路复用的模型处理接入的I/O请求，并且将之前同步阻塞调用后端服务的方式，改造成使用netty client（netty客户端）非阻塞调用的方式。改造之后，Netfix团队经过测试发现性能提升了20%左右。

除此之外，API网关中执行的动作有些是可以预先定义好的，比如黑白名单的设置，接口动态路由；有些则是需要业务方依据自身业务来定义。**所以，API网关的设计要注意扩展性，**也就是你可以随时在网关的执行链路上，增加一些逻辑，也可以随时下掉一些逻辑（也就是所谓的热插拔）。

所以一般来说，我们可以把每一个操作定义为一个filter（过滤器），然后使用“责任链模式”将这些filter串起来。责任链可以动态地组织这些filter，解耦filter之间的关系，无论是增加还是减少filter，都不会对其他的filter有任何的影响。

Zuul就是采用责任链模式，Zuul1中将filter定义为三类：pre routing filter（路由前过滤器）、routing filter（路由过滤器）和after routing filter（路由后过滤器）。每一个filter定义了执行的顺序，在filter注册时，会按照顺序插入到filter chain（过滤器链）中。这样Zuul在接收到请求时，就会按照顺序依次执行插入到filter chain中的filter了。



Zuul 的责任链示意图

另外还需要注意的一点是，为了提升网关对于请求的并行处理能力，我们一般会使用线程池来并行的执行请

求。**不过，这就带来一个问题：**如果商品服务出现问题，造成响应缓慢，那么调用商品服务的线程就会被阻塞无法释放，久而久之，线程池中的线程就会被商品服务所占据，那么其他服务也会受到级联的影响。因此，我们需要针对不同的服务做线程隔离，或者保护。**在我看来有两种思路：**

- 如果你后端的服务拆分得不多，可以针对不同的服务，采用不同的线程池，这样商品服务的故障就不会影响到支付服务和用户服务了；
- 在线程池内部可以针对不同的服务，甚至不同的接口做线程的保护。比如说，线程池的最大线程数是1000，那么可以给每个服务设置一个最多可以使用的配额。

一般来说，服务的执行时间应该在毫秒级别，线程被使用后会很快被释放，回到线程池给后续请求使用，同时处于执行中的线程数量不会很多，对服务或者接口设置线程的配额，不会影响到正常的执行。可是一旦发生故障，某个接口或者服务的响应时间变长，造成线程数暴涨，但是因为有配额的限制，也就不会影响到其他的接口或者服务了。

你在实际应用中也可以将这两种方式结合，比如说针对不同的服务使用不同的线程池，在线程池内部针对不同的接口设置配额。

以上就是实现API网关的一些关键的点，你如果要自研API网关服务的话可以参考借鉴。另外API网关也有很多开源的实现，目前使用比较广泛的有以下几个：

- Kong是在Nginx中运行的Lua程序。得益于Nginx的性能优势，Kong相比于其它的开源API网关来说，性能方面是最好的。由于大中型公司对于Nginx运维能力都比较强，所以选择Kong作为API网关，无论是在性能还是在运维的把控力上，都是比较好的选择；
- Zuul是Spring Cloud全家桶中的成员，如果你已经使用了Spring Cloud中的其他组件，那么也可以考虑使用Zuul和它们无缝集成。不过，Zuul1因为采用同步阻塞模型，所以在性能上并不是很高效，而Zuul2推出时间不长，难免会有坑。但是Zuul的代码简单易懂，可以很好的把控，并且你的系统的量级很可能达不到Netfix这样的级别，所以对于Java技术栈的团队，使用Zuul也是一个不错的选择；
- Tyk是一种Go语言实现的轻量级API网关，有着丰富的插件资源，对于Go语言栈的团队来说，也是一种不错的选择。

那么你要考虑的是，这些开源项目适不适合作为AIP网关供自己使用。而接下来，我以电商系统为例，带你将API网关引入我们的系统之中。

如何在你的系统中引入API网关呢?

目前为止，我们的电商系统已经经过了服务化改造，在服务层和客户端之间有一层薄薄的Web层，**这个Web层做的事情主要有两方面：**

一方面是对服务层接口数据的聚合。比如，商品详情页的接口，可能会聚合服务层中，获取商品信息、用户信息、店铺信息以及用户评论等多个服务接口的数据；

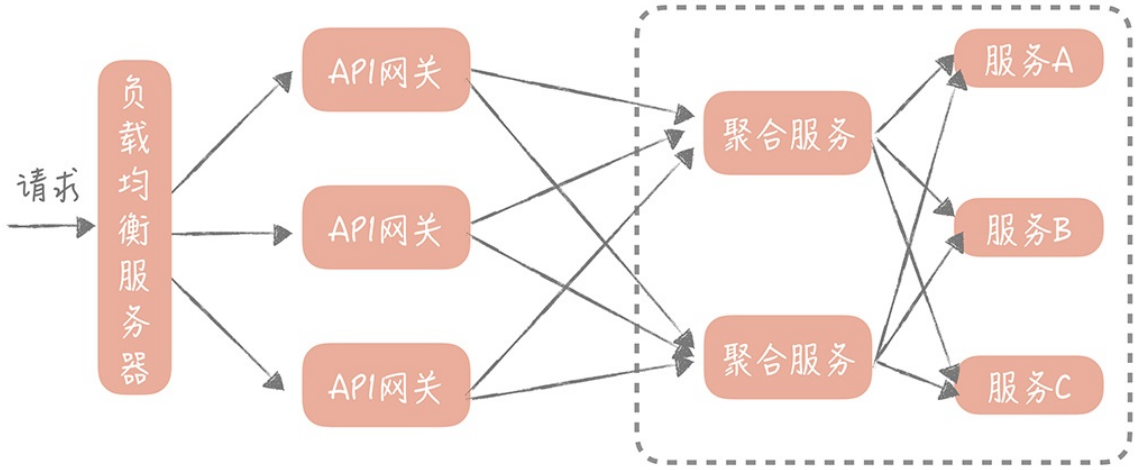
另一方面Web层还需要将HTTP请求转换为RPC请求，并且对前端的流量做一些限制，对于某些请求添加设备ID的黑名单等等。

因此，我们在做改造的时候，可以先将API网关从Web层中独立出来，将协议转换、限流、黑白名单等事情，挪到API网关中来处理，形成独立的入口网关层；

而针对服务接口数据聚合的操作，**一般有两种解决思路：**

1. 再独立出一组网关专门做服务聚合、超时控制方面的事情，我们一般把前一种网关叫做流量网关，后一种可以叫做业务网关；
2. 抽取独立的服务层，专门做接口聚合的操作。这样服务层就大概分为原子服务层和聚合服务层。

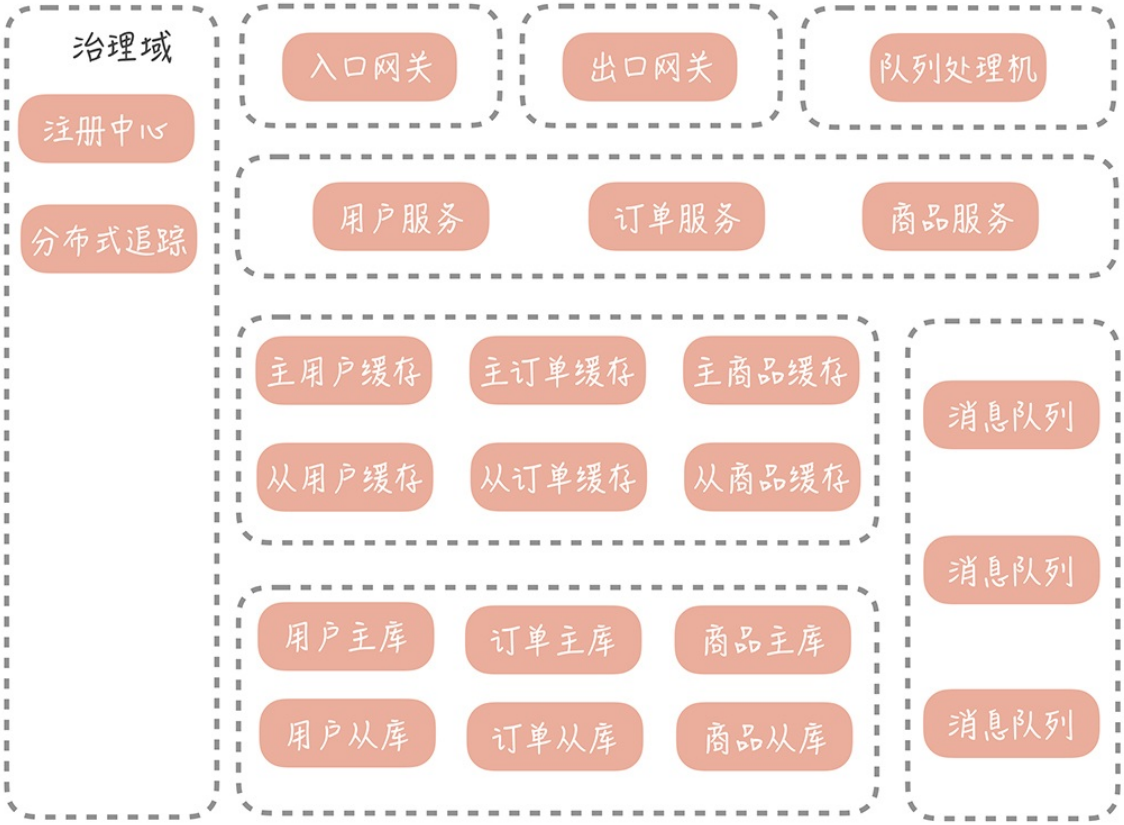
我认为，接口数据聚合是业务操作，与其放在通用的网关层来实现，不如放在更贴近业务的服务层来实现，**所以，我更倾向于第二种方案。**



网关部署示意图

同时，我们可以在系统和第三方支付服务，以及登陆服务之间部署出口网关服务。原先，你会在拆分出来的支付服务中，完成对于第三方支付接口所需要数据的加密、签名等操作，再调用第三方支付接口，完成支付请求。现在，你把对数据的加密、签名的操作放在出口网关中，这样一来，支付服务只需要调用出口网关的统一支付接口就可以了。

在引入了API网关之后，我们的系统架构就变成了下面这样：



系统架构图

课程小结

本节课我带你了解了API网关在系统中的作用，在实现中的一些关键的点，以及如何将API网关引入你的系统，**我想强调的重点如下：**

1. API网关分为入口网关和出口网关两类，入口网关作用很多，可以隔离客户端和微服务，从中提供协议转换、安全策略、认证、限流、熔断等功能。出口网关主要是为调用第三方服务提供统一的出口，在其中可以对调用外部的API做统一的认证、授权，审计以及访问控制；
2. API网关的实现重点在于性能和扩展性，你可以使用多路I/O复用模型和线程池并发处理，来提升网关性能，使用责任链模式来提升网关的扩展性；
3. API网关中的线程池，可以针对不同的接口或者服务做隔离和保护，这样可以提升网关的可用性；
4. API网关可以替代原本系统中的Web层，将Web层中的协议转换、认证、限流等功能挪入到API网关中，将服务聚合的逻辑下沉到服务层。

API网关可以为API的调用提供便捷，也可以为将一些服务治理的功能独立出来，达到复用的目的，虽然在性能上可能会有一些损耗，**但是一般来说，**使用成熟的开源API网关组件，这些损耗都是可以接受的。所以，当你的微服务系统越来越复杂时，你可以考虑使用API网关作为整体系统的门面。

一课一思

你的项目中是否有使用API网关呢？你在使用API网关的时候，遇到过什么样的问题吗？欢迎在留言区与我分享你的经验。

最后，感谢你的阅读，如果这篇文章让你有所收获，也欢迎你将它分享给更多的朋友。

精选留言：

- stubborn 2019-11-25 07:53:00
  请教一下老师，增加聚合服务有什么约束条件吗？比如聚合服务要同时写入服务A和服务B，这样很容易引起分布式事务。 [1赞]

- Julien 2019-11-25 07:16:53
  上一讲的负载均衡和这一讲的API网关是什么关系呢？ [1赞]

  作者回复2019-11-25 08:36:43
  作用不同，负载均衡作用是分流，API网关作用是服务治理

- 蓝魔丶 2019-11-26 11:07:43
  请教老师一下，如果有了性能很不错的入口网关的时候，还有必要使用nginx等web端代理工具嘛？

- 蓝魔丶 2019-11-26 10:56:50
  最后的服务蓝图中web层应该还是保留吧，把它放到网关和微服务之间，比如提到了聚合服务就是web层服务

- 张德 2019-11-25 21:11:22
  老师 请教一下 这个oauth2.0的登录认证 请求是先经过网关后验证是否登录了 还是什么别的流程 能否讲一下这个架构下 登录鉴权应该怎么做？？？

- 约书亚 2019-11-25 16:45:12
  我经历的网关变化挺复杂，最早是nginx + spring cloud zuul + 人工手写的BFF（BFF近似于聚合服务），其实这时候nginx意义不大
  后来做改造了spring cloud zuul部分功能，和BFF逐渐合并。
  目前是用traefik顶在前面做限流熔断鉴权服务发现等工作，后接具体微服务或者BFF，相当于又将网关功能拆分了。
  未来考虑去掉BFF中的Zuul换成其他更容易进行复杂编程的异步IO的网关，比如spring cloud gateway

- XD 2019-11-25 09:11:20
  用过kong，当时倒腾了一个很简陋的lua脚本做防刷，打算抽空看一下tyk

- 峰 2019-11-25 08:47:39
  多路同步复用在哪一节课讲的?

- 博弈 2019-11-25 08:26:39
  老师您好，我们最近项目中使用到了Spring cloud gateway，配置为使用https访问，在用jmeter进行压力测试时，1秒钟发送2000个请求，会出现ssl hand shake timeout错误，也在配置文件中设置了handshake timeout时间为60秒，还是会出现，请问老师有没有好的解决办法？谢谢

- 刘楠 2019-11-25 07:58:26
  好像一个过滤器，系统