Qing Gateway—高性能响应式云原生网关的设计与实现

一、需求分析

随着互联网与云计算的蓬勃发展，高流量、高并发的挑战与日俱增，极大促进了微服务架构的演变。分布式架构和微服务可以使后台服务变得高可用、加快开发周期并降低成本，但云原生应用程序在提供可靠连接方面带来了新挑战，对流量入口的并发性、无损性要求也越来越高。

因此，网关诞生了，现在市面上主流的网关有 spring cloud gateway、zuul、openrestry，对三者进行对比如下图 1-1 所示：

网关	限流	鉴权	监控	易用性	可维护性	成熟度
Spring Cloud Gateway	可以通过IP，用户，集群限流，提供了相应的接口进行扩展	普通鉴权、auth2.0	Gateway Metrics Filter	简单易用	spring系列可扩展强，易配置可维护性好	spring社区成熟，但gateway资源较少
Zuul2	可以通过配置文件配置集群限流和单服务器限流亦可通过filter实现限流扩展	filter中实现	filter中实现	参考资料较少	可维护性较差	开源不久，资料少
OpenResty	需要lua开发	需要lua开发	需要开发	简单易用，但是需要进行的lua开发很多	可维护性较差，将来需要维护大量lua脚本	很成熟资料很多

图 1-1 三种网关性能对比

可见，spring cloud gateway 要比其它两款网关综合性价比高。但是 spring cloud gateway 的缺点如下：需要引入第三方组件才能实现负载均衡和限流，如使用

Sentinel 组件进行熔断限流配置；只对开发人员友好，没有管理台界面，需要引入单独的监控组件查看网关负载等实时运行信息。Qing Gateway 正是为了解决这一系列问题。

Qing Gateway 是一个高性能、多协议、易扩展、分布式、响应式的 API 网关，拥有交互友好的管理平台界面。单机网关可承受 10w Qps 流量，网关处理损耗仅为 1-3ms，性能不输 Spring Cloud Gateway。兼容主流的微服务工具：Nacos、Redis、Mysql。基于 SPI 机制，支持热插拔，用户可以定制化开发，满足用户各种场景的现状和未来需求。

二、概要设计

2.1 总体设计

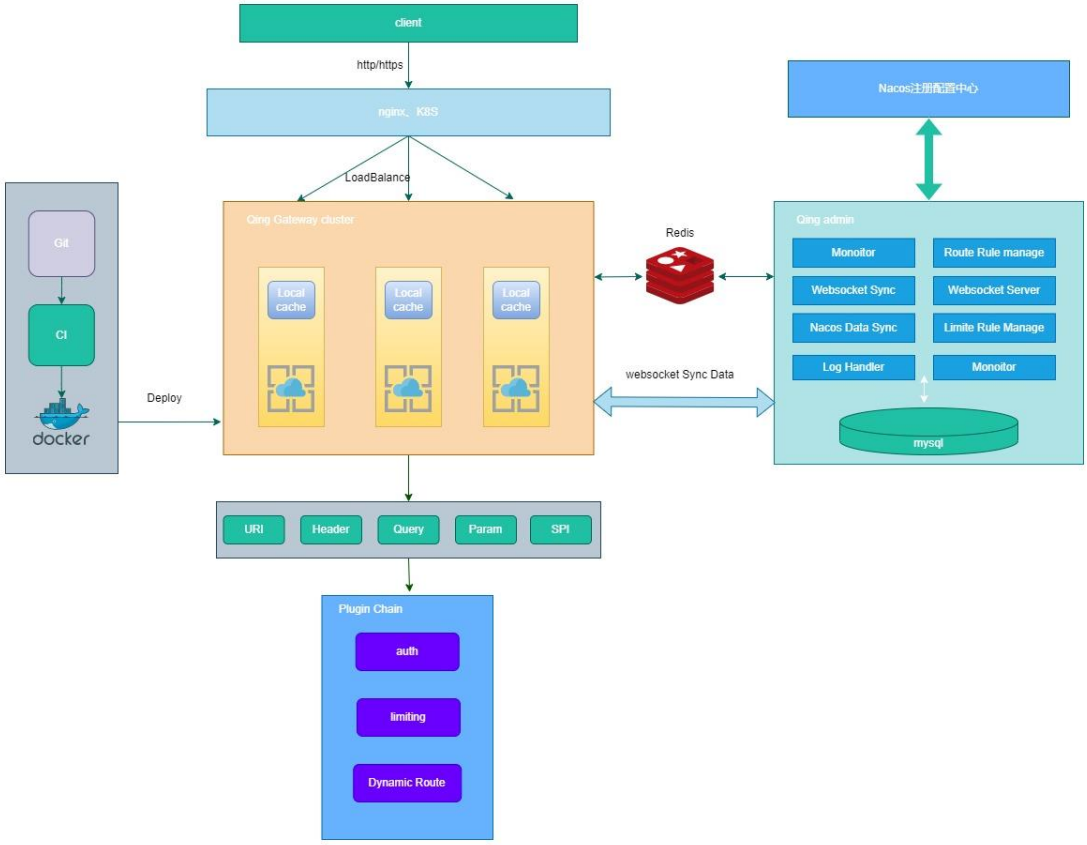

图 2-1 网关整体架构

网关是一个典型的 IO 密集型应用，因此传统的阻塞 IO（BIO）不适合高性能网关的构建，Qing Gateway Server 采用同步非阻塞的 I/O 模型（NIO）来处理高并发、大量连接的问题，对应的框架是 Spring Webflux；使用 Redis 作为消息队列、缓存；管理后台由 Springboot+React+Mysql 搭建；管理后台与网关 Server 之间采用 WebSocket 通信，管理后台统一接入多网关结点；选用 Nacos 作为微服务配置中心。

2.2 模块设计

项目主要分为两大模块：Qing Gateway server、Qing admin。

Qing Gateway server，为网关单节点的服务端，是网关逻辑的核心。分为以下几个模块介绍：

- 基于责任链设计模式、插件化设计思想，设计一条插件链，默认注册鉴权插件、限流插件、动态路由插件，除此之外，用户通过继承 AbstractQingPlugin 接口可自定义插件。
- 基于 SPI 机制，热插拔负载均衡规则。
- 缓存模块，将限流规则、路由规则、服务及实例规则等缓存至本地，使用时直接走缓存，减少网络 IO 次数。
- 数据同步模块是基于 Websocket，与 admin 中心建立长连接，实时接收 admin 的数据，并同步至本地缓存。

Qing admin，为集成所有网关结点的数据同步中枢、消息发送中枢。分为以下几个模块介绍：

- 注册中心与网关节点配置，可在管理台配置 Nacos 注册中心的地址、

各网关节点的 Websocket 通信的地址。

图 2-2 系统配置

- 动态路由配置，可根据路由断言+转发的目标服务+优先级动态添加路由匹配规则，热更新机制，无需重启。当请求来到网关时，Qing Gateway 将根据配置的负载均衡策略选择目标服务中的一个实例进行转发。

图 2-3 动态路由

- 服务实例配置，如图 2-4 所示，admin 可以对服务实例的协议、版本号、集群名、权重作热更新，同步至 Nacos。

图 2-4 服务实例配置

- 限流规则配置，支持对单账号、单接口进行分钟级、秒级限流熔断策略配置，如图 2-5 所示。

图 2-5 限流规则配置

- 实时监控中心，如图 2-6 所示，系统实时监控网关节点的 QPS、内存使用情况、JVM 使用情况，以便开发运维人员快速定位问题。

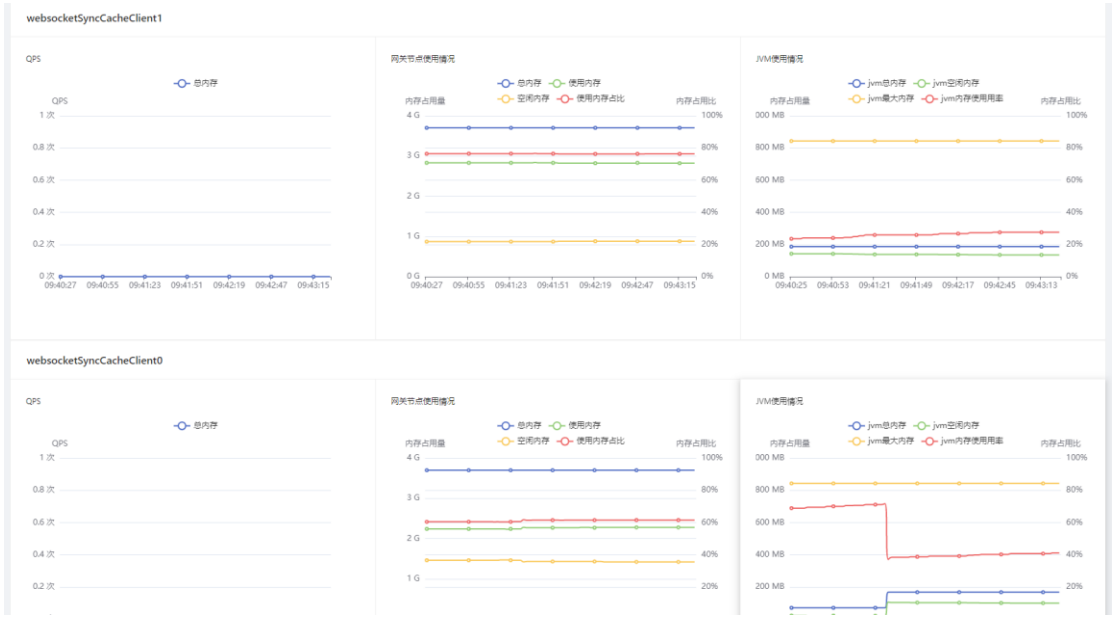

图 2-6 实时监控中心

- 数据同步模块，开启一个后台子线程去轮询拉取 Naocs 中心的在线服务及实例数据，开启一个线程池去维护各网关节点的 Websocket 通道。
- 事件侦听器，当 admin 有要通知网关节点的时刻，触发异步通知事件：限流事件、规则事件、服务及实例事件。

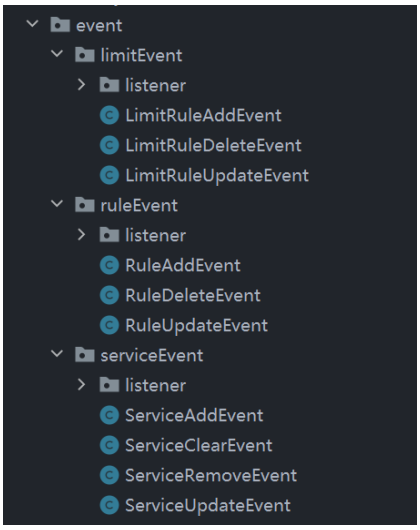

图 2-7 事件侦听

- 鉴权控制器，admin 项目采用 jwt token 作为身份校验工具，对密码采用 MD5 加密算法。
- 调用行为日志分析，如图 2-8 所示，可查看访问者源 IP、源请求路径、代理 URI、目标服务实例等信息。

源IP	源访问路径	代理uri	路由断言	目标服务	目标服务实例	创建时间
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8901/traffic/getTraffic	example1	traffic	124.222.224.173:8901	2022-04-27 11:54:07
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8901/traffic/getTraffic	example1	traffic	124.222.224.173:8901	2022-04-27 11:54:05
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8900/traffic/getTraffic	example1	traffic	124.222.224.173:8900	2022-04-27 11:54:04
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8900/traffic/getTraffic	example1	traffic	124.222.224.173:8900	2022-04-27 11:54:03
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8901/traffic/getTraffic	example1	traffic	124.222.224.173:8901	2022-04-27 11:54:01
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8900/traffic/getTraffic	example1	traffic	124.222.224.173:8900	2022-04-27 09:38:27
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8901/traffic/getTraffic	example1	traffic	124.222.224.173:8901	2022-04-27 09:38:26
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8901/traffic/getTraffic	example1	traffic	124.222.224.173:8901	2022-04-27 09:38:25
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8900/traffic/getTraffic	example1	traffic	124.222.224.173:8900	2022-04-27 09:38:24
127.0.0.1	/example1/traffic/getTraffic	http://124.222.224.173:8901/traffic/getTraffic	example1	traffic	124.222.224.173:8901	2022-04-27 09:38:04

< 1 2 3 4 5 ··· 37876 > 10 条/页

图 2-8 调取行为日志

- 负载均衡策略配置，通过 SPI 机制，负载均衡策略可热插拔配置。

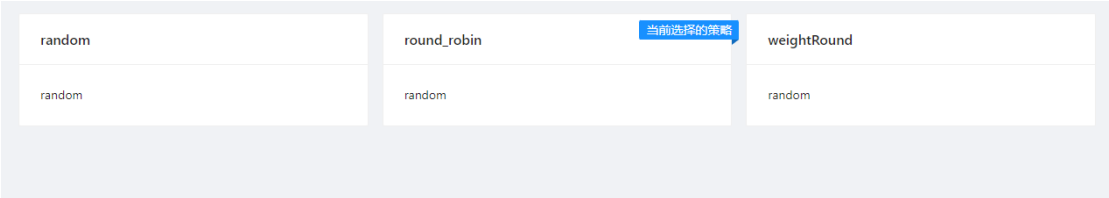

图 2-9 负载均衡配置

- 访问者 IP 热力图，通过调用百度地图开放 API，可以对日志中的调用者进行定位，最终处理的数据由前端展示出热力图，方便用户画像。

图 2-10 访问 IP 热力图

三、详细设计

3.1 插件链设计

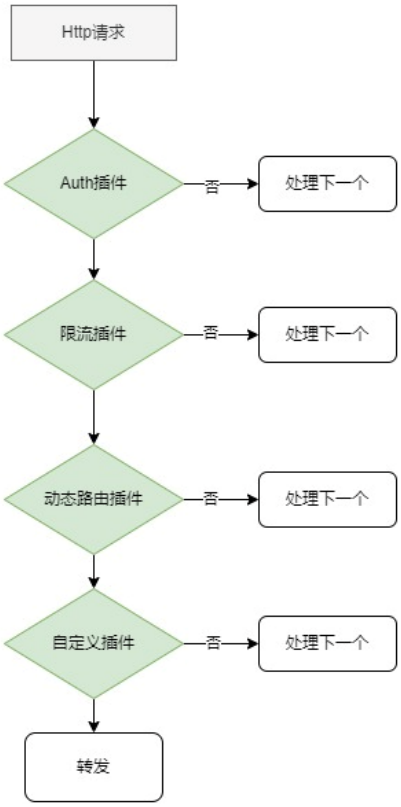

图 3-1 插件链

如图 3-1 所示，基于责任链设计模式和插件化思想，将网关的处理逻辑抽象成多个插件，按顺序去处理推到下一层或拒绝服务。

3.2 动态路由设计

网关使用 Nacos 作为微服务统一注册中心，因此路由规则主要有路由断言、服务名、优先级组成。网关会将匹配路由断言的请求，在目标服务的所有实例中根据配置的负载均衡策略选择一个实例转发。如下图 3-2 的规则，可以将 /example1/medical/getMedical 的服务转发至 medical 服务下。

图 3-2 路由规则

3.3 熔断限流设计

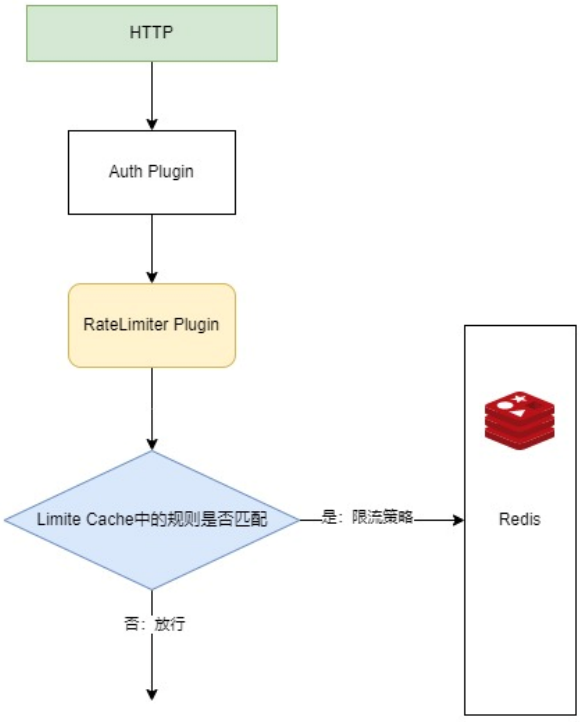

图 3-3 限流流程

网关目前基于 Redis，支持计数器限流、滑动窗口限流、漏桶限流、令牌桶限流四种策略。同样使用 SPI 热插拔技术，可动态配置。

3.4 SPI 扩展设计

SPI 全称为 Service Provider Interface，是 JDK 内置的一种服务提供发现功能，一种动态替换发现的机制。

目前支持负载均衡扩展、RateLimiter 扩展，用户分别实现 LoadBalance 与 RateLimiterAlgorithm 接口即可。

3.5 平滑发布设计

针对每个服务实例，可设置其版本号来完成平滑发布。如下图 3-4，当 124.222.224.173:8901 的服务版本升级为 2.0 后，网关将服务无缝衔接转发到该服务下的最高版本实例，随后将 1.0 版本的实例逐步升级至 2.0，完成平滑发布。

图 3-4 设置版本号

3.6 分布式数据同步设计

为了提升网关的性能，Qing Gateway 将所有的路由规则、限流规则、服务及实例数据缓存在 JVM 内存里面。在集群部署/分布式场景中，Qing Gateway 自主研发了一套将 Admin 控制台的数据，远程同步到每一个 Qing Gateway 网关节点 JVM 内存的方案，流程如图 3-5 所示。

Qing admin 开启一个子线程轮询拉取 Nacos 数据，首次启动时，将 Nacos 的

数据全量同步至网关节点，之后采用数据增量同步的策略，减少数据包的大小，加快消息传输速度。当在控制台更改路由规则、实例信息、限流规则时，会触发 Spring 的发布事件机制，异步将消息通知网关节点。

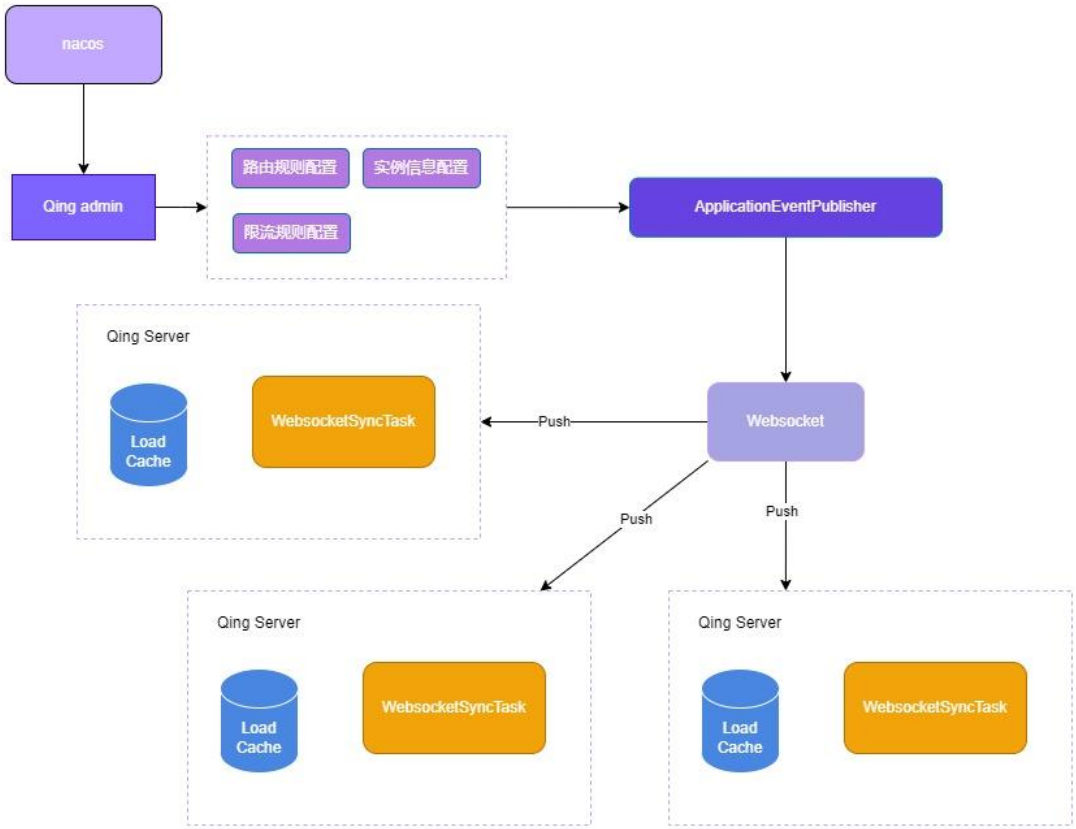

图 3-5 分布式数据同步方案

3.7 实时监控设计

如下图 3-6 所示，各个网关节点通过一个子线程每 1 秒查询自身 QPS、memory、JVM 运行状态，写入 Redis 中，管理台前端每 2s 进行一次数据查询，admin 后台将从 Redis 中读到的数据返回给前端。

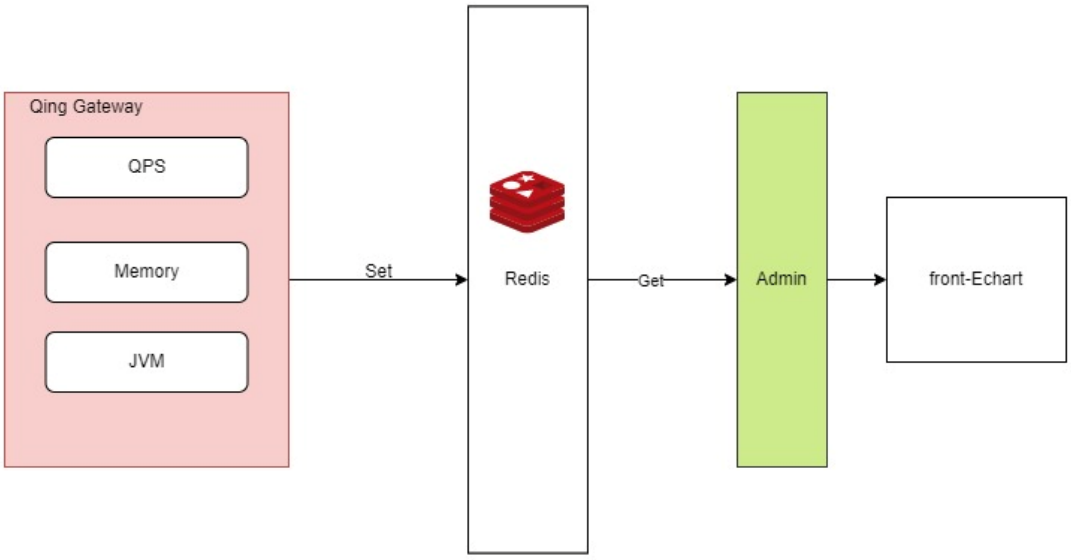

图 3-6 实时监控方案

四、测试报告

4.1 网关插件链损耗

经实验得，如图 4-1 所示，网关插件链的平均损耗是 1-3ms，对整个链路几乎无损。

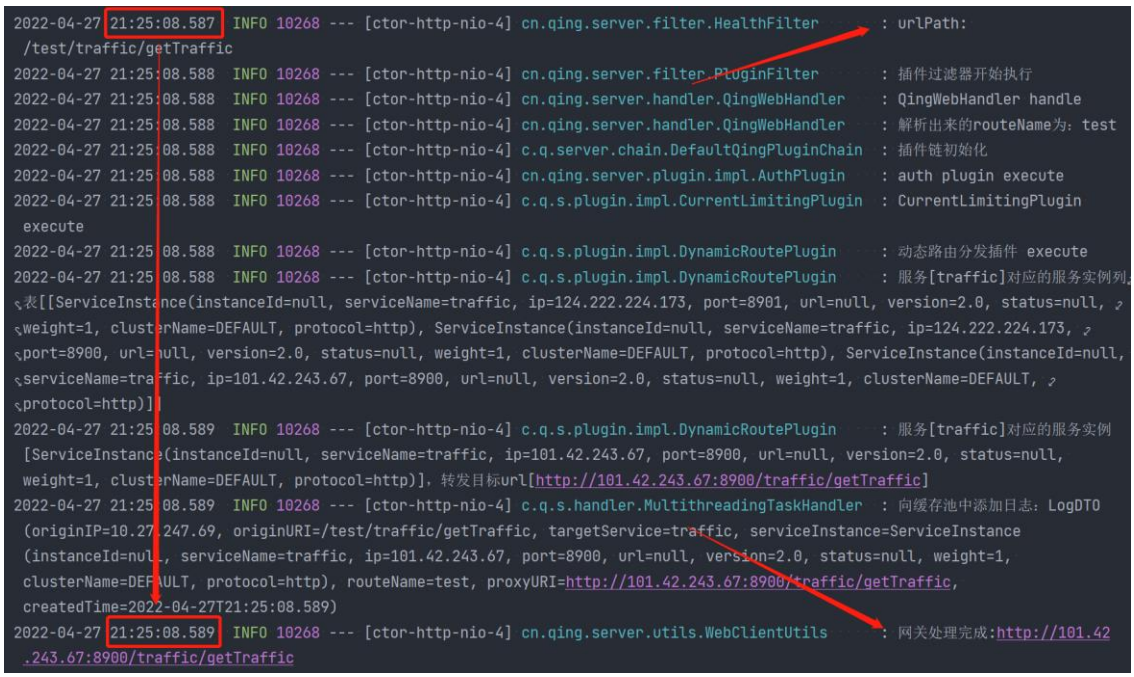

图 4-1 插件链损耗

4.2 单机网关性能测试

如图 4-2 所示，在 4 核 16G 的环境对网关单机测试，设置线程数为 1000，2 秒内发送，循环 200 次，也就是 10w 的 QPS，错误率为 0.13%，因此得出结论，网关单节点能承受 10w 的并发量。

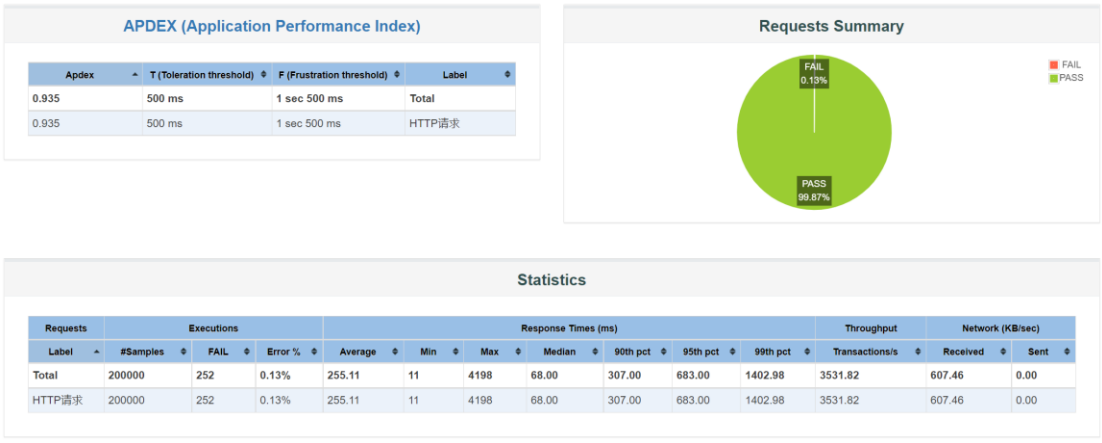

图 4-2 单机网关性能

4.3 网关集群性能测试

分别在一台 4 核、16G 的，三台 2 核 4G 的服务器搭建四个网关节点，使用 Nginx 按照 4，1，1，1 权重做负载均衡。如图 4-3 所示，设置线程数为 1000，2 秒内发送，循环 400 次，也就是 20w 的 QPS，错误率为 0.01%。因此得出结论，集群模式下，可承受至少 40w 的 QPS。

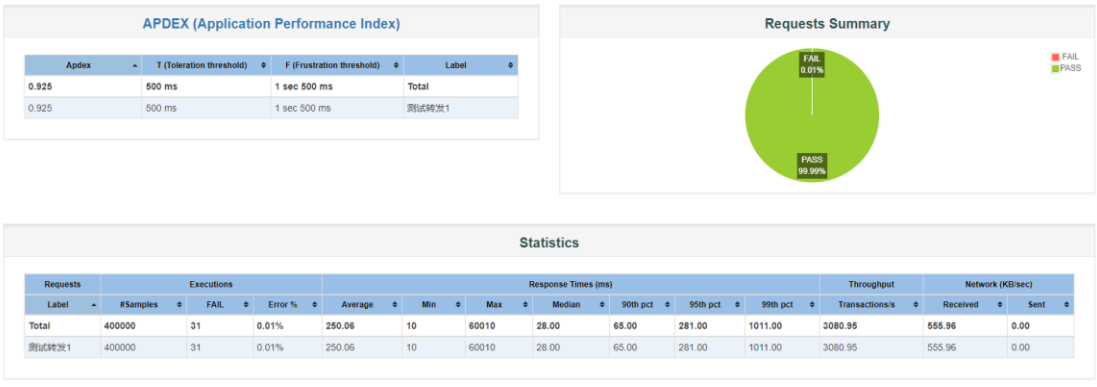

图 4-3 网关集群性能

五、使用及部署

5.1 本地启动

1. 下载代码:

```
git clone https://github.com/conghuhu/qing-gateway.git
cd qing-gateway
mvn clean install
```

2. 在 qing-admin 下的 application-dev 文件配置 Mysql 和 Redis 连接信息。
3. 在 qing-server 下的 application-dev 文件配置 Redis 连接信息。
4. 在 Mysql 中运行 qing_gateway.sql 文件。
5. 运行 qing-admin 下的 cn.qing.admin.QingAdminApplication 作为 admin。
6. 运行 qing-server 下的 cn.qing.server.QingServerApplication 作为网关节点 server。
7. 打开 localhost: 8080/static/index.html 就可以看到控制台界面，默认用户名和密码为 admin、123456。

5.2 docker 生产部署

待补充....

六、项目总结

本项目总结技术点如下:

- 运用多种设计模式：责任链模式、单例模式、工厂模式、侦听器模式，增强了代码的可读性、维护性和扩展性。

- 基于 NIO 异步非阻塞、响应式模式，增大了网关吞吐量。
- 将路由规则、服务实例数据等信息缓存至 JVM 内存中，使网关前期处理请求的损耗极低，仅 1-3ms。
- 针对日志处理、QPS 收集采用阻塞队列，合并处理的方式，减少网络 IO 次数，极大提高性能。
- 插件化的思想，增强了用户的自定义扩展性。
- 对访问者 IP 进行地图热力图处理，方便快速定位用户画像。

下一步计划如下：

- 接入 RPC 协议支持，如 Dubbo、gRpc。
- 面向 IOT 设备，接入 MQTT 协议支持。
- 基于容器化技术，引入动态扩容机制。
- 对 Netty 进行调优，使性能达到最佳。