恒丰银行分布式核心系统 -API 网关技术原型落地实践

恒丰银行科技开发部 赵毅 张涛

随着微服务架构的流行以及服务拆分粒度的精细化，RPC、分布式事务、服务注册 / 发现、负载均衡、集群容错、服务治理和 API 网关等概念逐渐成为了构建微服务架构的基础组件。

本文将结合恒丰银行分布式核心系统建设过程中的实施经验，就 API 网关的形成背景、需求概述、技术架构和设计亮点等方面展开论述。

一、形成背景

时至今日，很多银行出于系统、应用、数据等资源集中管理的目的，核心系统仍然采用单体式应用的部署和运行方式，并通过路由总控和应用总控进行交易接入、公共检查、安全控制、服务路由的管理。当然，很多银行也在逐步尝试分布式架构，但一般选择应用在核心系统以外的渠道系统或产品系统上，并通过 ESB 系统替代原先路由总控和应用总控的作用。

可以看到，本文谈及的 API 网关在定位上与前面提到的总控和 ESB 十分类似，都是作为后端服务为外围应用提供入口级服务，以消除不同应用之间的技术差异，让不同的应用服务协调运作，实现了不同服务之间的通信与整合。但在功能上，API 网关相对来说更注重稳定性和可用性，所以其设计模式更加轻量级，在确保接入安全有效、流量可控的前提下，专注于实现协议转换、报文适配、动态路由和异步通讯等基础功能。辅助用户简单、快速、低成本、低风险地实现微服务聚合、前后端分离、系统集成，有针对性地解决调用者与服务提供者之间错综复杂的调用关系，有助于加强后端服务 API 的管理。

二、需求概述

为了更好地设计适应银行业务的 API 网关产品，我们首先对业界现有的同类产品进行了调研，包括 Zuul、Kong、Tyk 三款。如图 1 所示，Zuul 主要面向 Netflix 公司自己使用，而 Kong 和 Tyk 更多地面向第三方付费企业，在功能上基本都以提供基础功能和扩展机制为主，主要包括安全认证、路由、流量控制和监控等。由此可见，这三款产品的受众群体基本一致，以面向互联网企业为主，主要应对高并发和网络攻击，与银行作为金融行业的定位还有一定差异。

就金融行业而言，其 API 网关建设原则应首先着眼于立足行业自身特点，即在保证可用性、稳定性和扩展性的基础上，重点加强功能性、安全性和可维护性等方面的建设。

如图 2 所示，恒丰银行 API 网关功能建设目标分为以下几点：

1.ARCHITECTURE（底层架构）

（1）平台无关性，基于自主研发的技术框架，不依赖其它第三方架构。

开源产品	主要功能
Zuul	基础身份认证、动态路由、静态响应处理、流量控制、金丝雀测试（按客户维度）、安全控制
Kong	HTTP 基本认证、CORS（跨域资源共享）、文件日志、流量控制、请求转发、nginx 监控
Tyk	基础身份验证、监控、负载均衡、缓存、报文转换、API 构建 / 测试功能

图 1 同业功能调研

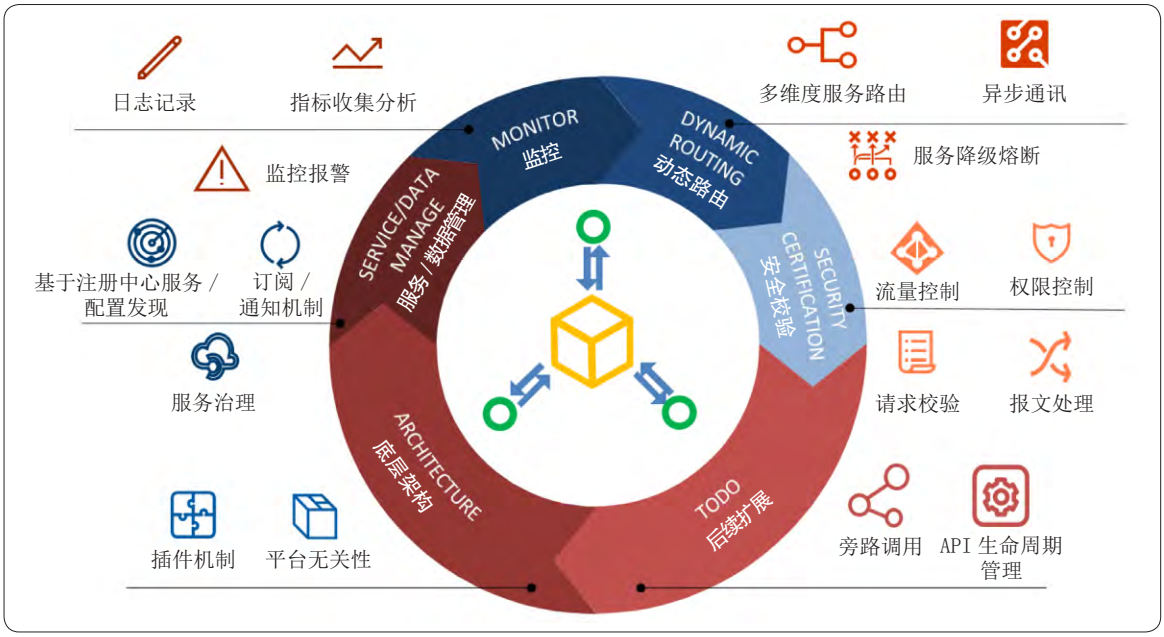

图 2 恒丰银行 API 网关功能建设目标

（2）插件机制，框架支持良好的功能扩展机制，便于后期维护升级。

2. SERVICE/DATA MANAGE（服务 / 数据管理）

（1）基于注册中心服务 / 配置发现，指通过类似于 Zookeeper 注册中心进行数据持久化，不依赖于数据库。

（2）订阅 / 通知机制，通过注册中心的订阅 / 通知机制实现数据的及时更新。

（3）服务治理，一方面通过业务数据的维护，控制请求流量及可访问的服务范围；另一方面通过注册中心及时感知后端服务 API 的状态变化。

3.SECURITY CERTIFICATION（安全校验）

（1）流量控制，指对请求流量进行控制的功能。

（2）权限控制，指对请求访问权限控制的功能。

（3）请求校验，指对请求信息的校验功能，一般为报文头中的公共信息。

（4）报文处理，报文转换、解析等附加功能。

4. DYNAMIC ROUTING（动态路由）

（1）多维度服务路由，通过多种维度路由规则配置实现的服务识别能力。

（2）异步通讯，通过异步通讯机制实现的高效交互能力。

（3）服务降级熔断，通过服务隔离，实现自动化服务降级和断路的能力。

5. MONITOR（监控）

（1）日志记录，按照平台统一规范的日志记录行为。

（2）指标收集分析，应用侧通过日志收集 agent 向流数据平台提供日志数据，并由流数据平台收集、加工和分析的过程。

（3）监控报警，流数据平台根据监控规则甄别、输出报警信息的过程。

6. TODO（后续扩展）

（1）旁路调用，交易流程中需要外调其它系统，且交易结果不敏感、无需同步处理的部分，由 API 网关基于异步机制直接外调的场景。

（2）API 生命周期管理，实现后端服务 API 发布、测试、调试、下线和属性维护等全生命周期的管理。

三、技术架构

在明确了功能清单后，还需进一步明确技术架构。首先是技术选型的过程，仍然是以 Zuul、Kong、Tyk 三个业界流行的框架做为依据。

如图 3 所示，Kong 和 Tyk 使用的开发语言与我行现有的技术体系相差较大，存在一定的学习成本，以至于我们在这方面的技术储备存在不足，无法充分预估将会出现的问题和隐患；除此之外，Kong 必须构建在 Cassandra 或 PostgreSQL 数据库上，而 Tyk 不支持 SOAP 格式。比较而言，Zuul 凭借 Netflix 公司强大的技术背景，无论在开发语言、运行机制还是在配套产品等方面都更符合我行的特点和目标。所以，我行分布式核心系统 API 网关以 Zuul 作为技术原型，基于 Filter 机制以及 PRE-ROUTING-POST-ERROR 模型（PRPE「前正后反」模型）实现，在框架层面为集成的标准化进行了铺垫，同时支持良好的扩展，如图 4 所示。

在系统架构方面，如图 5 所示，API 网关作为分布式技术框架中的一个组成部分，通过责任链机制（FilterChain）和 Java SPI 机制实现业务功能的插件化管理，并借助于注册中心模块（Registry）实现服务 / 配

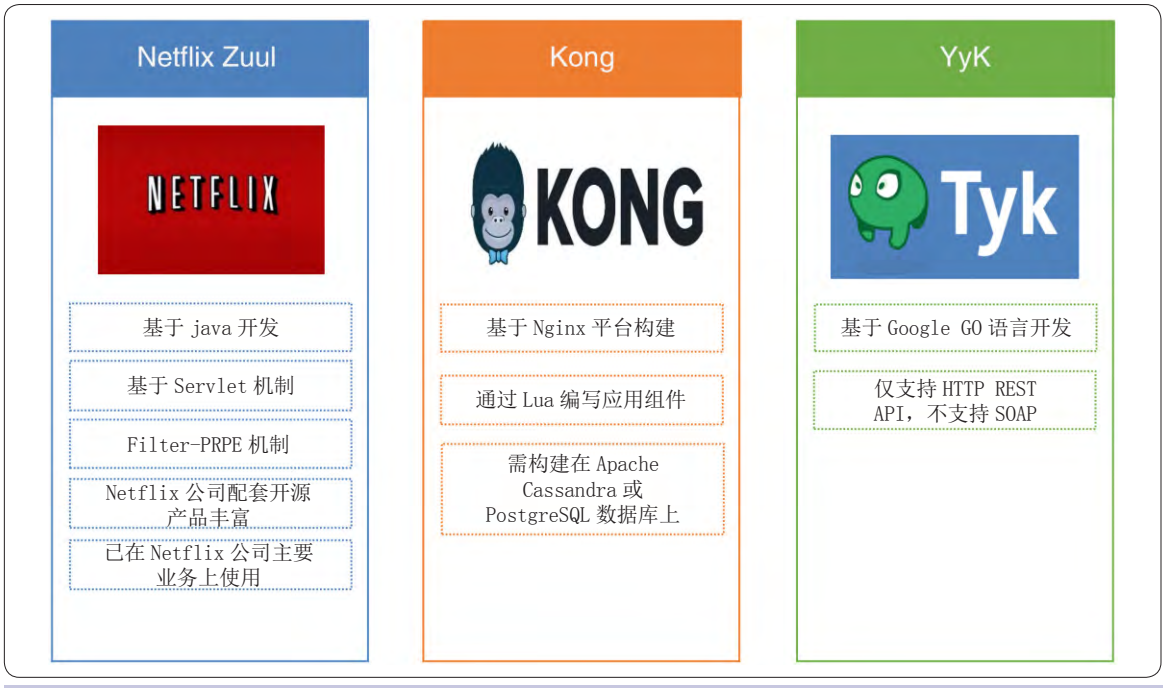

图 3 同业技术调研

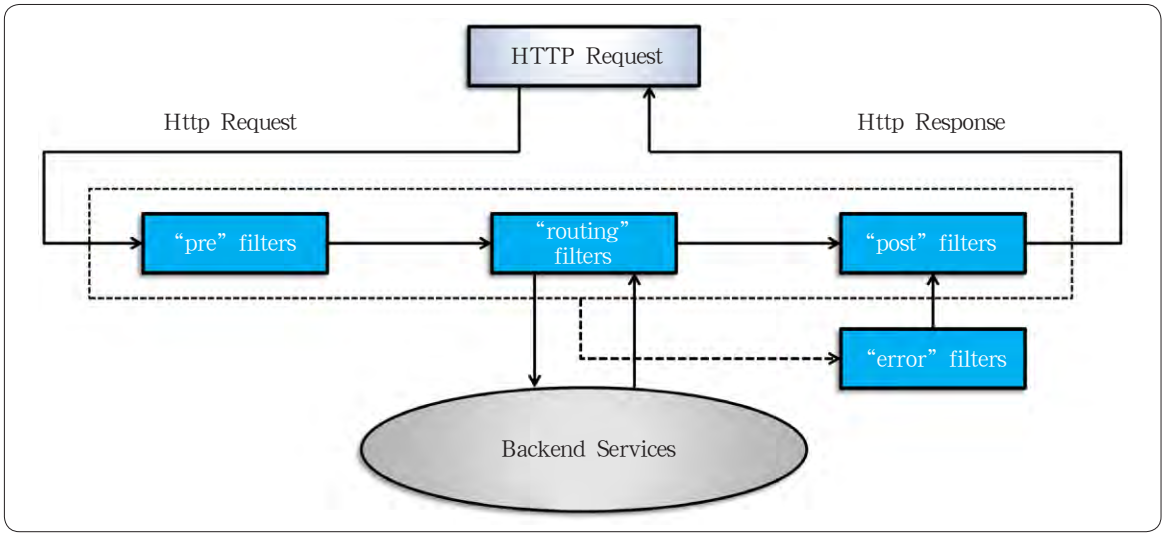

图 4 filter–PRPE 机制

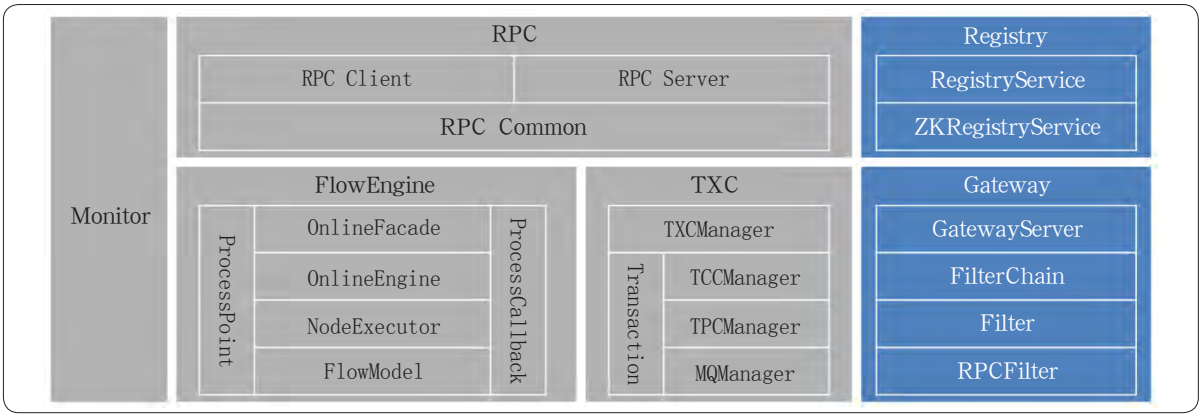

图 5 系统架构

置信息的高效管理。

前面提到了我行 API 网关的建设要基于金融行业的特点，尤其注重功能性和安全性，这主要体现在功能部分提到的服务/配置管理、安全校验、动态路由几个方面，而驱动和促使这些功能发挥预期效果的是与之相对应的数据结构设计。

如图 6 所示，网关涉及的数据主要包括基础配置信息和业务配置信息，其中，基础配置信息以文件的形式保存在本地，业务信息则通过注册中心进行订阅 / 通知，目前提供了基于 Zookeeper 的注册中心机制。具体来说，基础配置信息提供了网关启动所需的环境配置信息，包括注册中心配置、外调后端服务配置以及默认的基础规则配置，包括 IP 黑白名单模式、报文头要素列表等，用来设定业务功能开关及校验键值对的键值。业务配置信息则提供了网关运行所需的业务规则配置，它是通过管理端进行维护的，包括 IP 黑白名单配置、请求报文头配置、报文解析规则配置和路由规则配置，这些信息都是以键值对的形式存储，保障了模型的通用性、扩展性和可移植性。除此之外，还可以通过管理端动态维护基础规则配置，以覆盖配置文件中的默认值。

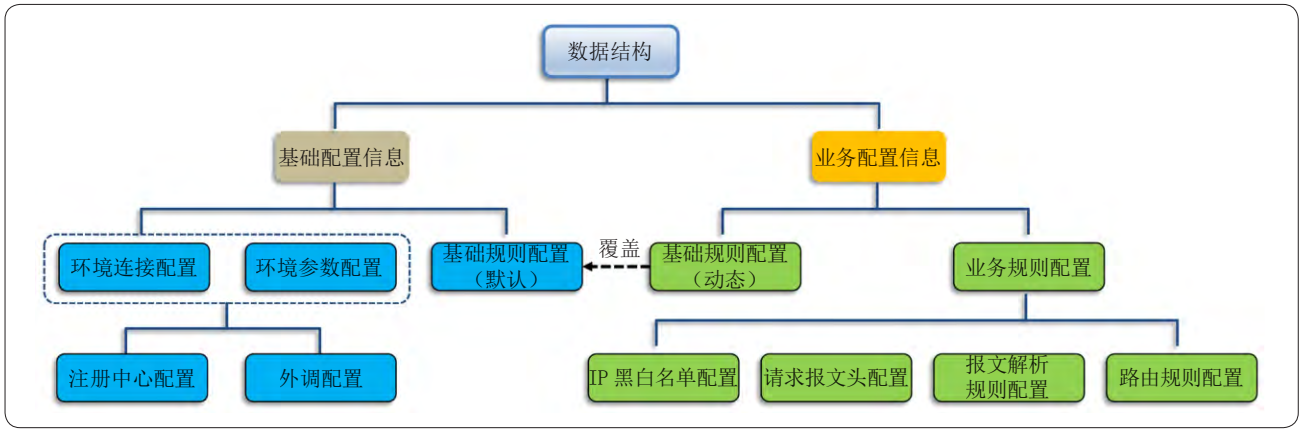

图 6 数据结构设计

四、设计亮点

在介绍了整体技术架构后，下面针对设计中的技术亮点进行逐一说明。

1. 平台无关性

API 网关因其轻量级、功能独立和业务侵入低等特性，在设计时，我们便考虑以 Java 原生的机制实现，包括通过 SPI 机制实现 filter 及其它扩展点的扩展加载机制，以及基于第三方工具类实现的部分业务功能，例如基于 StAX 实现的 xml 报文向 map 格式的转换。除此之外，整个架构不依赖于其他第三方平台或框架，实现了平台无关性。图 7 展示了网关的内部逻辑。

2. Filter–PRPE 机制

基于 Zuul 的实现原理，我们在此之上进行了改

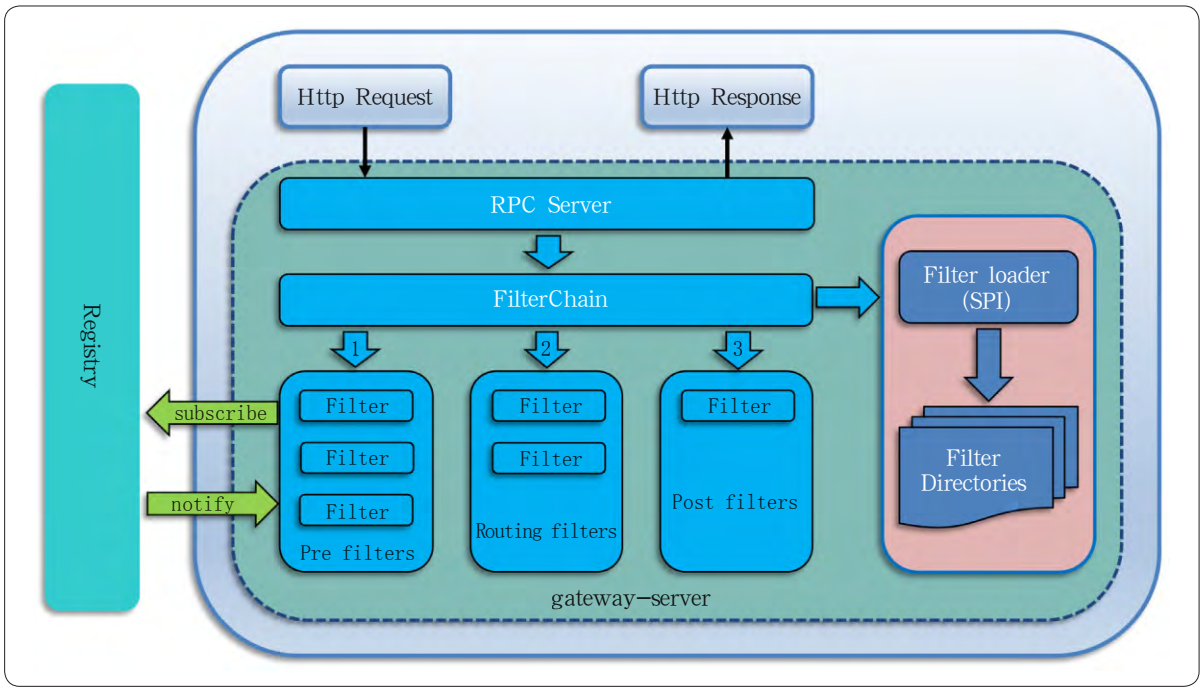

图 7 API 网关内部逻辑

良。首先，在网关启动时会依次调用每个 filter 的 init 方法，以通过注册中心获取运行期所需数据并缓存在内存中，以提高运行期效率；其次，接口类中定义了 doPre 和 doPost 方法，以实现 filter 的双向机制，即通过 FilterChain 按序依次执行完每个 filter 的 doPre 方法后，再反序执行重载了 doPost 方法的 filter，其适用于在 doPre 中预留了资源，需要在 doPost 中释放的场景，如图 8 所示。

3. 服务 / 配置数据动态管理

区别于其他 API 网关依赖数据库进行数据持久化的做法，我们选择通过更为高效和去中心化的注册中心进行数据管理。初始业务数据是通过管理端进行的维护，在网关启动时便会订阅这些数据，以便数据的任何更改都会及时通知网关以更新缓存，如图 9 所示。

4. 请求过滤机制

请求过滤机制属于 filter 中的 PRE 类别，其作用是

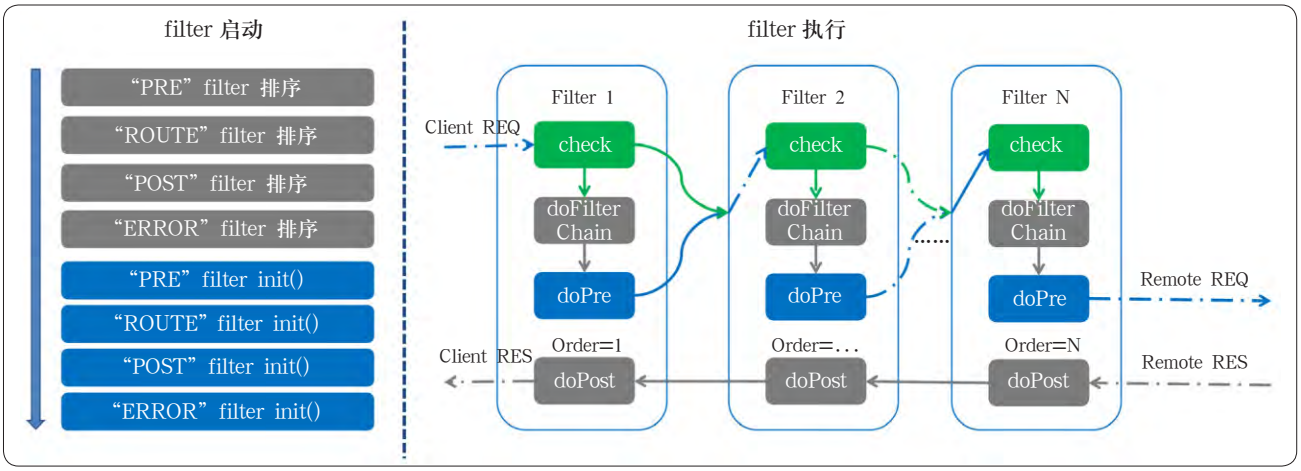

图 8 Filter–PRPE 实现

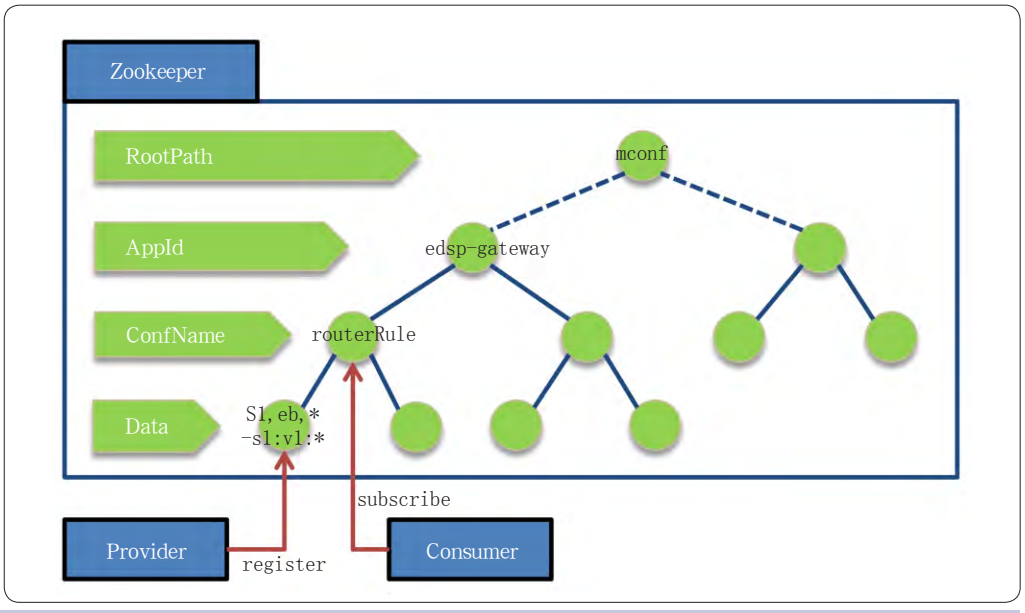

图 9 请求过滤机制

对请求信息进行基础检查或处理，目前提供了报文转换、报文解析、黑白名单检查和请求参数校验功能。基于前面第 3 点提到的数据动态管理功能，每个功能都支持按需动态启 / 停，同时支持动态扩展，如图 10 所示。

5. 多维度动态路由机制

路由机制属于 filter 中的 ROUTING 类别，也是所有 filter 中最为重要的核心的模块，其作用是根据请求报文中的参数，将交易发送到与之匹配的后端服务，也是真正体现“网关”理念的模块。目前该模块提供了路由规则解析 / 检查和服务外调两个基础功能。其中，路由规则解析 / 检查通过图 11 所示的数据模型，实现了多维度的路由规则控制。

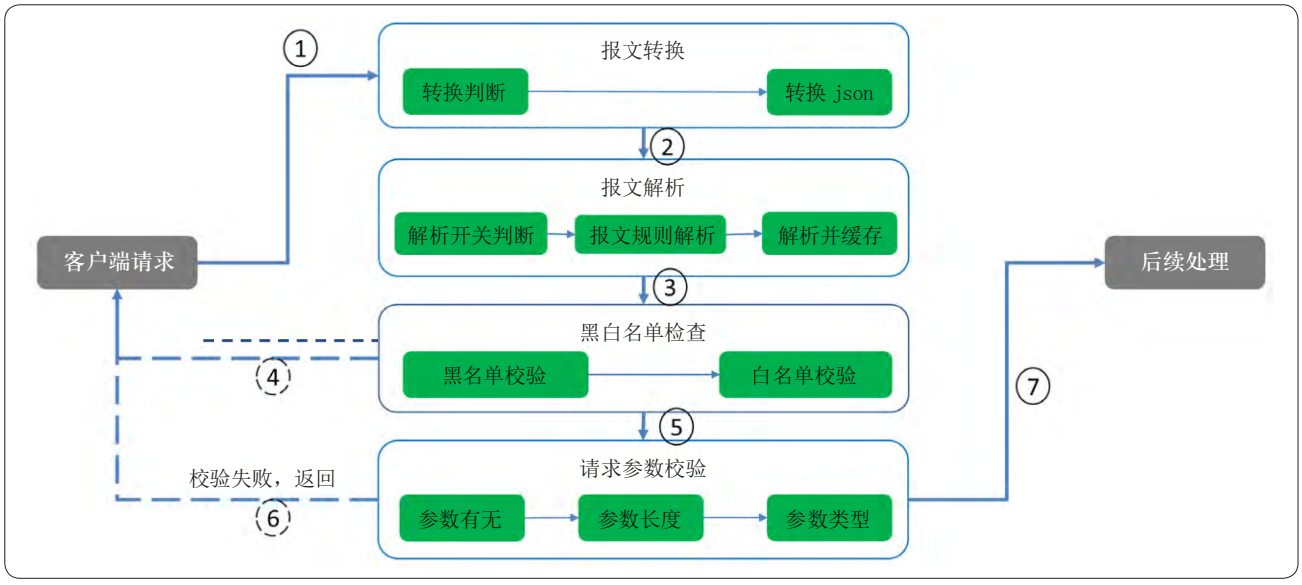

图 10 请求过滤机制

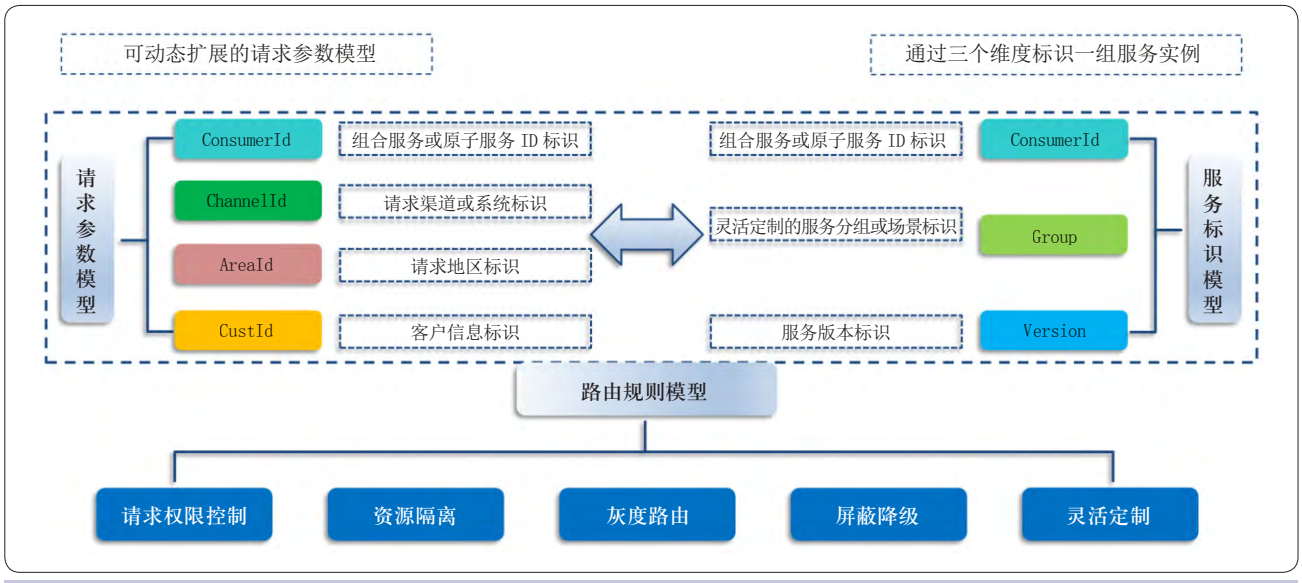

图 11 路由规则模型

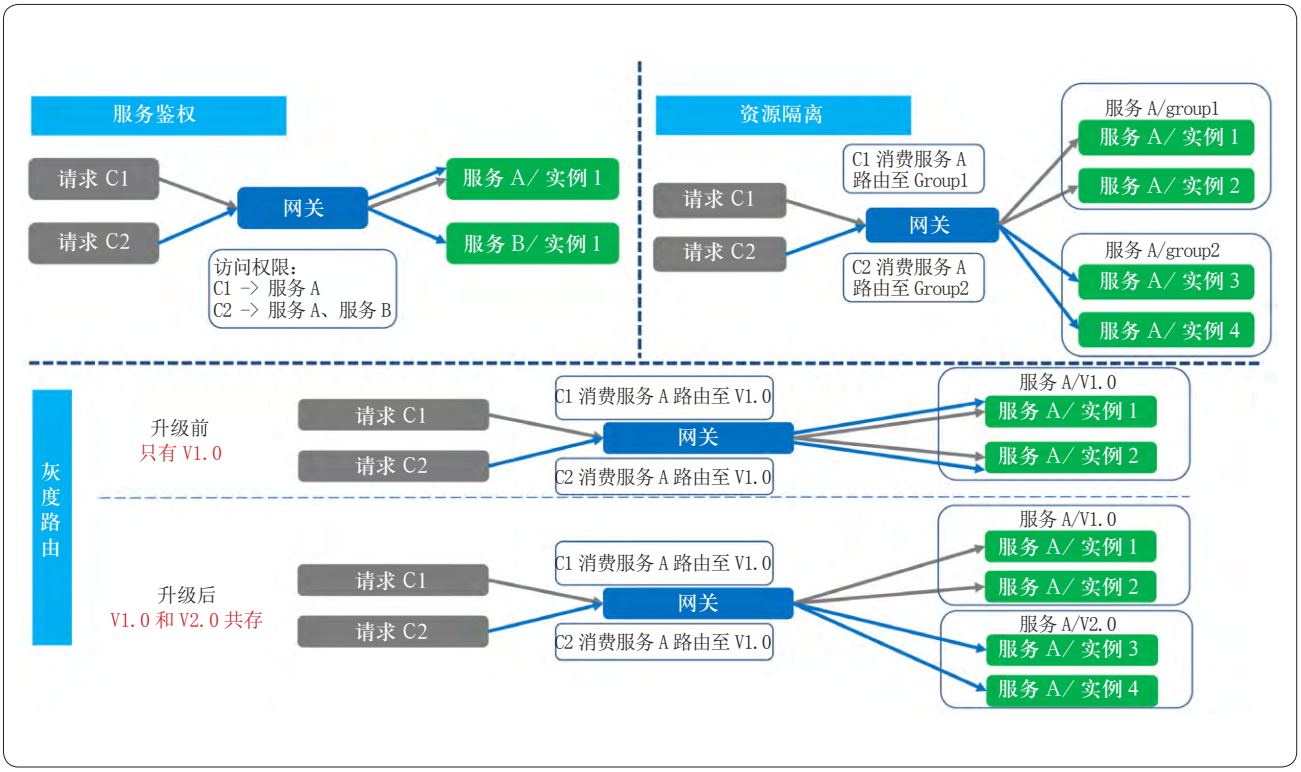

图 12 多维度路由机制

后端服务为增强其扩展性和部署灵活性，通过交易码（ConsumerId）、场景标识（Group）和版本号（Version）三个维度进行管理，而前端请求则可以根据业务需要灵活定制，在 ConsumerId 之外还可以增加渠道标识（ChannelId）、地区标识（AreaId）和客户信息标识（CustId）等，通过这两部分信息的规则映射，以实现请求权限控制、资源隔离、灰度路由和屏蔽降级等功能，如图 12 所示。

6. 服务降级 / 熔断机制

分布式系统随着功能数量、访问规模的增加，即使在可以通过水平扩展提升性能的情况下，仍然会存在个别服务失效而引发的连锁反应，最终导致灾难性的整体性问题。

通过引入 Netflix 公司的 Hystrix 机制，通过资源隔离机制实现了如下目标：

（1）防止单个依赖耗尽容器内所有用户线程。

（2）降低系统负载，对无法及时处理的请求快速失败（failfast）而不是排队。

（3）提供失败回退，以在必要时让失效对用户透明化。

（4）使用隔离机制（例如“舱壁”/“泳道”模式，“熔断器”模式等），降低依赖服务对整个系统的影响。

任何新系统、新架构和新平台的尝试都不会是一帆风顺的，我们也只是在分布式微服务架构的建设之路上迈出了一小步，但正是这点尝试，让我们切身感受到了微服务的魅力所在，也坚定了我们继续探索新知识的决心。FCC