基于 OpenResty 的高性能 API 网关的设计与实现

黄章洁　刘晓娟　关展鹏
（广东白云学院，广东 广州 510450）

摘要：API 网关是微服务架构的关键组件，互联网企业需要轻巧、云原生友好的 API 网关，以满足特定的性能场景需求。本文介绍了一款基于 OpenResty 的高性能 API 网关，它在设计上创新地采用了令牌桶算法，有效应对高并发场景下的性能挑战。与传统的漏桶算法相比，此 API 网关在处理大量请求时展现出更高的吞吐量和更优的性能。本文详细介绍了该网关的设计原理、架构，以及实现方式，并验证其在现实应用中的潜力和优势。通过对比分析，本文展示了该网关在高并发处理方面超越传统方法的能力，为高流量网络应用提供了一种更为高效和可靠的解决方案。

关键词：API 网关；OpenResty；令牌桶算法；高并发

在当前互联网技术的发展和普及背景下，API（应用程序编程接口）在连接不同服务和平台的作用上愈发关键且不可忽视。API 通过将企业的业务逻辑封装起来，并对外提供统一的接口，不仅使得应用程序客户端能够有效地与后端服务进行交互，而且在协调整个系统的请求流量和服务访问方面扮演着“门户”的角色。

OpenResty 是一个强大的 Web 平台，它结合了 Nginx 服务器和 Lua 脚本语言，提供了高性能和高度可定制的服务处理能力。特别地，本文介绍的 API 网关采用了令牌桶算法，这一算法在高并发环境下相较于传统的漏桶算法，展现出更优越的处理能力和更高的吞吐量。本文将展示该网关在高并发处理方面超越传统方法的能力，强调其在微服务架构中的应用价值和前景。

一、背景

随着 API 的使用越来越广泛，其性能和可用性问题变得尤为显著。尽管闭源的商业产品提供了全面的功能，但对于许多企业而言，它们往往伴随着平台锁定和二次开发困难等问题。开源 API 网关因其灵活性和可定制性而受到青睐，但这些解决方案也有自身的不足，例如配置生效慢、插件不支持热加载等问题。因此，探索一种既能提升性能又能保持高度可用性和灵活性的 API 网关，成为了业界的焦点。

本文在 OpenResty 的基础上，通过算法的改进和架构的微调，进一步提升 API 网关的性能和可靠性。这不仅能满足当前对高效处理大量 API 请求的需求，还能提供更加适应云原生环境和开发者友好的解决方案。

在传统 API 网关的设计中，常见的流量控制算法如漏桶算法，虽然能够有效地限制数据的传输速率，保证网络的稳定性，但它在高并发场景下存在明显的局限性。漏桶算法通过固定速率“漏出”请求，对突发流量的处理能力有限，这在大规模并发请求的情况下会导致处理延迟，甚至服务不可用。此外，该算法的固定速率处理机制在动态变化的网络流量环境下缺乏灵活性，不能根据实际情况自动调整，这限制了其在快速变化的网络条件和需求下的应用。此外，许多传统 API 网关依赖于复杂的配置和管理，这在云原生环境中可能导致灵活性和可扩展性的问题。随着云计算和微服务架构的普及，API 网关需要更快的配置更新和更高效的资源管理，传统方法往往难以满足这些要求。因此，开发一款能够有效应对高并发场景，同时具备高度灵活性和易于配置的 API 网关，对于提升互联网应用的整体性能和用户体验至关重要。本项目所开发的这款 API 正是在这样的背景下应运而生，它不仅在处理高并发请求方面具有优势，还能提供更加灵活和云原生友好的解决方案。

二、API 网关的设计与实现

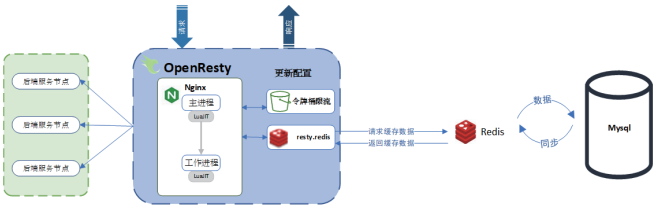
（一）基于 OpenResty 的架构：

OpenResty 平台被选为构建高性能 API 网关的基础，主要因其结合了 Nginx 的高效性和 Lua 脚本语言的灵活性。Nginx 作为一个轻量级的高性能 Web 服务器，它的事件驱动模型特别适合处理高并发请求，这对于 API 网关来说至关重要。

OpenResty 继承了 Nginx 的多进程架构，每一个工作进程都是复制主进程而得到的，其实，主进程中的 LuaJIT 虚拟机也会一起复制过来。在同一个工作进程内的所有协程，都会共享这个 LuaJIT 虚拟机，Lua 代码的执行也是在这个虚拟机中完成的。而在同一个时间点上，每个工作进程只能处理一个用户的请求，也就是只有一个协程在运行，用同步的代码逻辑实现非阻塞的调用。

OpenResty 通过整合 LuaJIT，允许在 Nginx 服务器上运行 Lua 脚本，为 API 网关提供了高度的可定制性和扩展性。这意味着 API 网关不仅能够高效处理请求，还能灵活地根据业务需求调整和扩展功能。

（二）网关架构设计



高性能网关架构设计

所有外部请求到达后端服务系统之前都会先经过网关处理，网关架构设计图如图 1 所示，从设计图可知，网关主要实现限流和缓存两个功能点。

限流，也称流量控制。是指系统在面临高并发，或者大流量请求的情况下，限制新的请求对系统的访问，从而保证系统的稳定性。限流会导致部分用户请求处理不及时或者被拒，这就影响了用户体验。所以一般需要在系统稳定和用户体验之间平衡一下。在这款 API 网关中，采用了令牌桶算法进行流量控制和管理。为了得到更好的限流效果，在限流之前需要对后端系统服务能力做好评估。后端系统本身的服务能力决定了使用令牌桶算法中全局令牌的数量。与传统的漏桶算法相比，令牌桶算法更加灵活地处理突发流量。它通过一个填充令牌的“桶”来控制数据的传输，允许在网络流量较低时积累令牌，然后在高流量时消耗这些令牌以处理更多请求。这种机制使得 API 网关能够更有效地应对流量波动，尤其是在高并发的使用场景下，能够显著提升网关的吞吐量和响应速度。

高性能网关会根据后端系统的服务能力初始化令牌桶中令牌的数量，这个数量是可以操作增加或者是减少的。考虑到用户首

次登录后端服务系统，不会携带 JSESSIONID，会临时把令牌发给用户发起请求所用的 IP，并把这个 IP 存储到 Redis 中。而每个用户后续发送的每个请求都会携带同一个 JSESSIONID 作为会话 ID，网关会把之前发到 IP 上的令牌传递给 JSESSIONID，并把这个 JSESSIONID 缓存到 Redis 中，并把原先存储的 IP 从 Redis 中删除。那么，当用户发起其他请求时，网关都要判断 Redis 中是否已经缓存这个 JSESSIONID，如果是的话网关直接放行，就不需要再让令牌数量减一。在 Redis 里面的 JSESSIONID 会配置一个有效期，比如十分钟。如果十分钟内网关再未收到携带这个 JSESSIONID 的请求，那这个有效期一旦过期的话，Redis 会自动把这个 JSESSIONID 给它删除。如果十分钟内网关仍旧能收到携带这个 JSESSIONID 的请求，那么会重置 JSESSIONID 的有效时间。因此需要一个定时函数，每隔一段时间，比如说一分钟两分钟之后，查询 Redis 里面已经缓存的 JSESSIONID 的数量。这样当前可用令牌的数量就等于最大的令牌数减去 Redis 里面已经缓存的 JSESSIONID 的数量，目前还不能够实时更新当前可用令牌数量。

（三）Redis 数据库的应用：

当网关需要应对高并发访问的情况时，一瞬间成千上万的请求到来，需要系统在极短的时间内完成成千上万次的读 / 写操作，这个时候的性能需求往往不是传统关系型数据库能够承受的。因此在设计网关架构时，添加了非关系型数据库 Redis 作为后端系统数据库的缓存，可以极大地缩短请求的处理时延。

resty.redis 是一个基于 cosocketAPI 的为 ngx_lua 模块提供 Luaredis 客户端的驱动，因此基于 OpenResty 的 API 网关能够通过 Lua 脚本快速地实现 Redis 客户端，确保处理性能不受影响。此外，这款 API 网关还支持微服务架构中的服务发现和动态配置更新，使得它能够无缝地融入云原生环境。在本次网关架构中，Redis 主要承担 2 类数据缓存，一类是后端系统服务过程中经常需要用到的业务数据，相当于把 Redis 作为 MySQL 等关系型数据的缓存。这样当用户需要从后端服务中访问这些业务数据时，就不必真的从后端系统服务中获取，而是网关从 Redis 中处理之后直接返回用户，可极大的减少用户请求的时延。令一类是网关本身需要使用的数据，例如上文提到的用户请求中携带的 JSESSIONID 的缓存，这部分数据与后端服务系统业务无直接关联，不需要考虑与后端 MySQL 数据库的同步。

三、应用案例和性能评估

（一）应用场景

本文选取了一个开源的“高校选课系统”作为验证高性能 API 网关能力的具体应用场景。高校选课系统是一个复杂且具有综合性的信息系统，涉及多类用户（如学生、教师和管理员）和各种类型的课程，典型地体现了复杂性、多样性、高负载和安全性等特点。它不仅是大学生熟悉的系统，更是验证 API 网关性能和可靠性的理想场景。

（二）性能评估和对比分析

为了全面评估基于 OpenResty 的 API 网关的性能，特别是在高并发处理方面的表现，我们设计了以下两步压力测试方案：

测试阶段	测试目的	测试方法	性能指标	结果分析
【阶段 1】测试选课系统本身的服务能力	确定无网关时选课系统的最大并发处理能力和相应的时延。	在无网关状态下逐步增加并发请求，直至时延明显上升。记录不同并发度下的响应时间和资源消耗。	最大并发数、响应时延、系统资源消耗（如 CPU、内存使用率）。	根据时延变化和资源消耗评估系统本身的性能极限。
【阶段 2】测试网关保护下的服务能力（使用令牌桶算法的网关）	评估令牌桶算法下 API 网关保护的选课系统性能表现。	在 API 网关使用令牌桶算法时，从第一步测试的最大并发度开始逐步增加并发请求，观察时延和系统资源消耗。	响应时延、系统资源消耗、并发处理能力。	评估令牌桶算法下的性能表现和系统稳定性。

在完成上述两步测试后，我们进行以下性能对比分析：对比无网关和有网关情况下的平均响应时间。

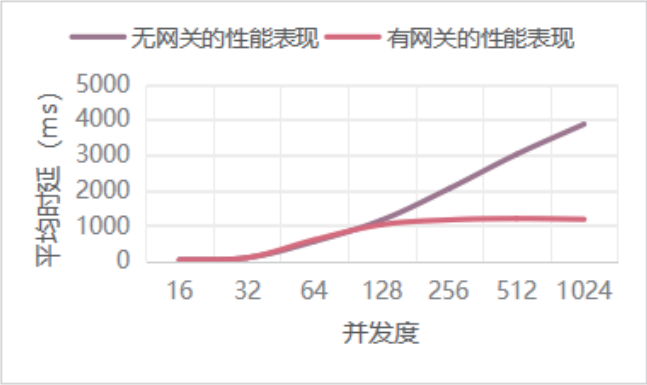


图 2 平均响应时间对比

基于上述测试结果，总结 API 网关在高并发处理方面的性能表现，特别是其在吞吐量、响应时间和系统稳定性方面相较于无网关情况的优势。

四、总结与展望

本文介绍的基于 OpenResty 的高性能 API 网关，在多个方面展现了创新性。最显著的创新点是采用了令牌桶算法，这不仅大幅提高了处理高并发请求的能力，也提升了网关的整体性能和效率。此外，该 API 网关结合了 Nginx 的高性能和 Lua 脚本的灵活性，使其成为一个高度可定制且响应迅速的解决方案。这些特性使其在现代的微服务架构和云原生环境中尤为适用。在微服务架构中，本文所实现的 API 网关的应用潜力尤为显著。它不仅能够有效管理和路由微服务之间的交互，还提供了必要的安全保护和流量控制功能。随着云原生应用的普及和微服务架构的日益复杂，这款 API 网关的灵活性和高性能特性将越来越受到企业和开发者的青睐。展望未来，随着技术的不断进步和市场需求的变化，基于 OpenResty 的 API 网关将继续追求更高的性能和更强的可靠性。我们预见到，该网关将通过不断的技术创新，如引入人工智能和机器学习算法来预测和管理流量，以及通过更加自动化和智能化的运维工具来简化管理过程，从而更好地满足未来复杂网络环境和业务需求的挑战。

参考文献：

[1] 金海峰，余傲雪 .Nginx 负载均衡器集群架构的实践应用 .[J]. 安徽电子信息职业技术学院学报，2023，22（03）：1-6.

[2] 董庆杰 . 高并发场景下软件架构的实现与优化 .[J]. 长江信息通信，2023，36（12）：94-96+99.

基金项目：广东省大学生创新创业训练计划资助项目（S202310822012）

基于 OpenResty 的高性能 API 网关的设计与实现

黄章洁　刘晓娟　关展鹏
（广东白云学院，广东 广州 510450）

摘要：API 网关是微服务架构的关键组件，互联网企业需要轻巧、云原生友好的 API 网关，以满足特定的性能场景需求。本文介绍了一款基于 OpenResty 的高性能 API 网关，它在设计上创新地采用了令牌桶算法，有效应对高并发场景下的性能挑战。与传统的漏桶算法相比，此 API 网关在处理大量请求时展现出更高的吞吐量和更优的性能。本文详细介绍了该网关的设计原理、架构，以及实现方式，并验证其在现实应用中的潜力和优势。通过对比分析，本文展示了该网关在高并发处理方面超越传统方法的能力，为高流量网络应用提供了一种更为高效和可靠的解决方案。

关键词：API 网关；OpenResty；令牌桶算法；高并发

在当前互联网技术的发展和普及背景下，API（应用程序编程接口）在连接不同服务和平台的作用上愈发关键且不可忽视。API 通过将企业的业务逻辑封装起来，并对外提供统一的接口，不仅使得应用程序客户端能够有效地与后端服务进行交互，而且在协调整个系统的请求流量和服务访问方面扮演着“门户”的角色。

OpenResty 是一个强大的 Web 平台，它结合了 Nginx 服务器和 Lua 脚本语言，提供了高性能和高度可定制的服务处理能力。特别地，本文介绍的 API 网关采用了令牌桶算法，这一算法在高并发环境下相较于传统的漏桶算法，展现出更优越的处理能力和更高的吞吐量。本文将展示该网关在高并发处理方面超越传统方法的能力，强调其在微服务架构中的应用价值和前景。

一、背景

随着 API 的使用越来越广泛，其性能和可用性问题变得尤为显著。尽管闭源的商业产品提供了全面的功能，但对于许多企业而言，它们往往伴随着平台锁定和二次开发困难等问题。开源 API 网关因其灵活性和可定制性而受到青睐，但这些解决方案也有自身的不足，例如配置生效慢、插件不支持热加载等问题。因此，探索一种既能提升性能又能保持高度可用性和灵活性的 API 网关，成为了业界的焦点。

本文在 OpenResty 的基础上，通过算法的改进和架构的微调，进一步提升 API 网关的性能和可靠性。这不仅能满足当前对高效处理大量 API 请求的需求，还能提供更加适应云原生环境和开发者友好的解决方案。

在传统 API 网关的设计中，常见的流量控制算法如漏桶算法，虽然能够有效地限制数据的传输速率，保证网络的稳定性，但它在高并发场景下存在明显的局限性。漏桶算法通过固定速率“漏出”请求，对突发流量的处理能力有限，这在大规模并发请求的情况下会导致处理延迟，甚至服务不可用。此外，该算法的固定速率处理机制在动态变化的网络流量环境下缺乏灵活性，不能根据实际情况自动调整，这限制了其在快速变化的网络条件和需求下的应用。此外，许多传统 API 网关依赖于复杂的配置和管理，这在云原生环境中可能导致灵活性和可扩展性的问题。随着云计算和微服务架构的普及，API 网关需要更快的配置更新和更高效的资源管理，传统方法往往难以满足这些要求。因此，开发一款能够有效应对高并发场景，同时具备高度灵活性和易于配置的 API 网关，对于提升互联网应用的整体性能和用户体验至关重要。本项目所开发的这款 API 正是在这样的背景下应运而生，它不仅在处理高并发请求方面具有优势，还能提供更加灵活和云原生友好的解决方案。

二、API 网关的设计与实现

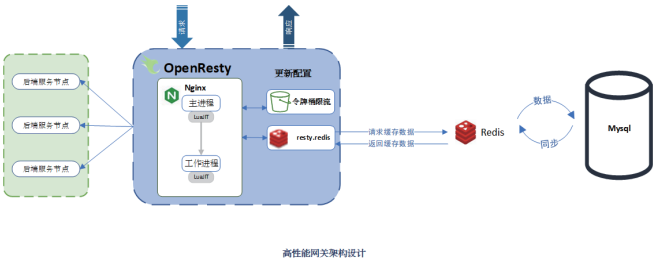
（一）基于 OpenResty 的架构：

OpenResty 平台被选为构建高性能 API 网关的基础，主要因其结合了 Nginx 的高效性和 Lua 脚本语言的灵活性。Nginx 作为一个轻量级的高性能 Web 服务器，它的事件驱动模型特别适合处理高并发请求，这对于 API 网关来说至关重要。

OpenResty 继承了 Nginx 的多进程架构，每一个工作进程都是复制主进程而得到的，其实，主进程中的 LuaJIT 虚拟机也会一起复制过来。在同一个工作进程内的所有协程，都会共享这个 LuaJIT 虚拟机，Lua 代码的执行也是在这个虚拟机中完成的。而在同一个时间点上，每个工作进程只能处理一个用户的请求，也就是只有一个协程在运行，用同步的代码逻辑实现非阻塞的调用。

OpenResty 通过整合 LuaJIT，允许在 Nginx 服务器上运行 Lua 脚本，为 API 网关提供了高度的可定制性和扩展性。这意味着 API 网关不仅能够高效处理请求，还能灵活地根据业务需求调整和扩展功能。

（二）网关架构设计



高性能网关架构设计

图 1 网关架构设计

所有外部请求到达后端服务系统之前都会先经过网关处理，网关架构设计图如图 1 所示，从设计图可知，网关主要实现限流和缓存两个功能点。

限流，也称流量控制。是指系统在面临高并发，或者大流量请求的情况下，限制新的请求对系统的访问，从而保证系统的稳定性。限流会导致部分用户请求处理不及时或者被拒，这就影响了用户体验。所以一般需要在系统稳定和用户体验之间平衡一下。在这款 API 网关中，采用了令牌桶算法进行流量控制和管理。为了得到更好的限流效果，在限流之前需要对后端系统服务能力做好评估。后端系统本身的服务能力决定了使用令牌桶算法中全局令牌的数量。与传统的漏桶算法相比，令牌桶算法更加灵活地处理突发流量。它通过一个填充令牌的“桶”来控制数据的传输，允许在网络流量较低时积累令牌，然后在高流量时消耗这些令牌以处理更多请求。这种机制使得 API 网关能够更有效地应对流量波动，尤其是在高并发的使用场景下，能够显著提升网关的吞吐量和响应速度。

高性能网关会根据后端系统的服务能力初始化令牌桶中令牌的数量，这个数量是可以操作增加或者是减少的。考虑到用户首