基于 KONG 的 API 集成系统的设计与实现

张 虎,张秋萍
(齐鲁工业大学(山东省科学院)山东省计算中心(国家超级计算济南中心)
山东省计算机网络重点实验室,山东 济南 250000)

摘 要:当前云计算平台中往往会包含若干个子系统,各个子系统要求的 API 格式是不一样的,而且有些子系统的 API 格式完全不符合 Restful API 格式。这就造成了云计算平台需要分别调用各个子系统的 API,且每个系统的 API 调用需要做个性化的处理,无形中增加了开发人员的开发时间和学习成本,同时也增加了云计算平台的运营和维护成本。针对这一问题,基于接口网关 KONG 设计并实现了一套 API 集成系统。该 API 集成系统会转发云计算平台中的所有请求,并在转发请求的过程中会将不符合 Restful 格式的 API 转成 Restful 格式的 API,方便云计算平台用统一的方式来调用不同系统的 API。测试结果表明,API 集成系统可以完成 API 集成的功能,将非 Restful 格式的 API 转为 Restful 格式。

关键词:云计算平台;接口网关;KONG;API;Restful
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2019)08-0102-05
doi:10.3969/j.issn.1673-629X.2019.08.020

Design and Realization of KONG-based API Integration System

ZHANG Hu,ZHANG Qiu-ping
(Shandong Provincial Key Laboratory of Computer Networks,Qilu University of Technology (Shandong Academy of Sciences),Shandong Computer Science Center (National Supercomputer Center in Jinan),Jinan 250000,China)

Abstract:The current cloud computing platform often contains several subsystems,each subsystem requires different API format,and some subsystem API format is completely not in line with the Restful API format. This is why the cloud computing platform needs to call the API of each subsystem separately,and the API call of each system needs to be personalized,which increases the development time and learning cost of developers virtually,and also increases the operation and maintenance cost of the cloud computing platform. In order to solve this problem,through the interface gateway KONG,a set of API integration system is designed and implemented. The API integration system will forward all the requests in the cloud computing platform,and in the process of forwarding the requests,the API that does not conform to the Restful format will be converted to the Restful format API,making it convenient for the cloud computing platform to call the APIs of different systems in a unified way. The test shows that the system can complete the API integration function and convert the non-restful API to Restful format.

Key words:cloud computing platform;interface gateway;KONG;API;Restful

1 概 述

随着云计算平台[1]功能的不断丰富,用于支撑云计算平台功能的子系统也在不断增加。当前的云平台支持物理资源的虚拟化[2]功能、应用的自动化部署功能、物理资源的监控功能、消息队列功能、对象存储[3]功能、短信邮件通知功能等等。

这些功能往往是通过不同的子系统来完成的。例如,物理资源的虚拟化功能由 Openstack[4]系统支撑,应用的自动化部署由 Cloudfoundry 系统支撑,物理资源的监控功能由 Zabbix[5]系统支撑,消息队列功能由 RabbitMQ[6]系统支撑,对象存储功能由 Swift[7]系统支撑,短信和邮件通知功能分别由腾讯云服务支撑。

这只是列举了云计算平台的常用功能,就已经涉及到这么多的子系统,而且每一个子系统都有其独立的 API 系统,这就导致了云计算平台需要分别调用不同子系统的 API。平台架构如图 1 所示。

收稿日期:2018-09-03 **修回日期**:2019-01-04 **网络出版时间**:2019-03-27
基金项目:山东省重点研发计划项目(2017GGX10130)
作者简介:张 虎(1987-),男,硕士,助理研究员,研究方向为云计算、大数据平台研发及应用。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20190327.1624.036.html

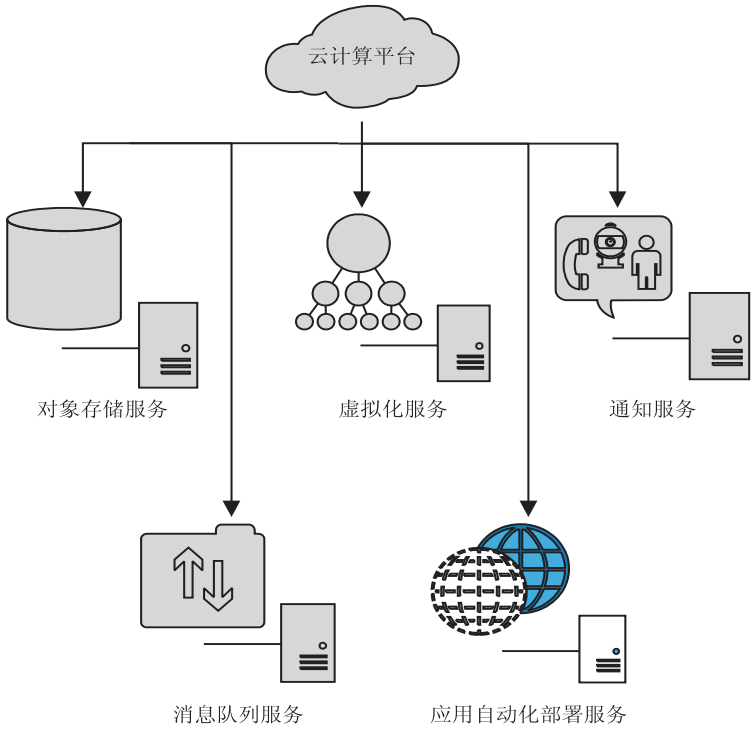

图 1　传统云计算平台框架

由于子系统的开发团队和开发风格的不同,导致了每个子系统的 API 格式也不相同。如果由云计算平台来调用不同子系统,不同格式的 API,不仅增加了云计算平台的开发工作量,也为云计算平台的日常维护和排除增加了困难。因此,如何使用统一的方式来调用不同系统的 API 是亟待需要解决的问题。

针对上述需求,文中提出了通过接口网关 Kong 来集成不同子系统 API 的方法。具体来说,KONG 统一接收云计算平台发送给不同子系统的请求,通过 url 路径的不同来转发给不同的子系统。在转发的过程中还会根据每个子系统的不同,转换请求的格式,以匹配不同子系统的 API 格式。通过这种方式,能够实现云计算平台使用统一的格式调用不同子系统的 API。加入 API 集成系统后的云计算平台架构,如图 2 所示。

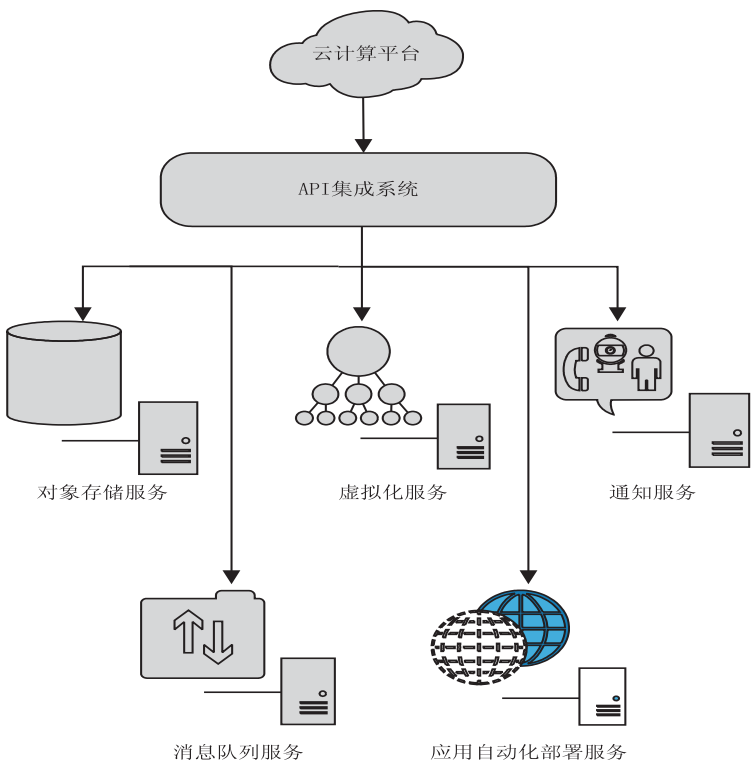

图 2　改进云计算平台框架

文中描述的 API 集成系统,不仅适用于云计算平台,而且适用于任何多子系统调用的平台。

2 接口网关 Kong 简介

Kong 是 Mashape 公司的开源 API 网关。Kong 是在客户端和(微)服务间转发 API 通信的 API 网关。其底层是由 NGINX[8] 驱动的,所以性能方面与 NGINX 相同。

Kong 原生支持主流数据库的读写操作,包括 Cassandra[9]、PostgreSQL[10] 等。Kong 可以将用户的自定义路径与访问路径写入数据库,在转发请求时动态改变请求的目的地址。这有利于用户灵活地添加转发地址,并且可以灵活地设计不同功能的插件。而 NGINX 只能通过配置文件的方式来完成这些功能,而且配置好之后需要重启 NGINX 的服务,这是很多长期服役的服务所不能接受的。

Kong 支持多节点的集群[11],这可以使 Kong 支持更多的访问流量,同时保证极低的网络延时。一个集群中的所有节点配置都是相同的,并且共用一个数据库。出于性能的原因,在转发请求时避免频繁数据库的访问,集群中的每个节点都会在本地缓存一部分数据,并且缓存的数据也是每个节点都相同的。

Kong 具有模块性:它可以与新的插件协同运行,扩展基本功能。可将 API 与许多不同的插件进行整合,以增强安全、分析、验证、日志及/或监测机制。并且 Kong 的插件可以访问 Kong 系统的数据库。本 API 接口集成系统,就是使用 Kong 的这一特性,开发出不同 API 格式转换的插件,并且将这些插件分别绑定到不同子系统的 API 接口上,实现统一格式调用不同格式 API 的功能。

目前很多公司都在使用 Kong,例如 CISO、Aol、Intel、Expedia 等。本 API 集成系统也是基于开源版 KONG 设计和实现的。

3 API 集成系统

文中主要介绍 API 集成系统的工作方式,并将 API 集成系统应用于集成 Openstack 和 Zabbix 接口。其中 Openstack 是主流的 IaaS[12] 开源管理平台;而 Zabbix 则是功能强大的主机监控平台。这两个平台应用广泛,而且两个平台的接口格式不同。Openstack 接口是标准的 RestfulAPI[13] 接口,而 Zabbix 系统的 API 接口则采用的是 JSON-RPC 2.0[14] 规范,所以通过 API 集成系统集成这两个平台的接口能够很好地说明 API 集成系统的工作方式和优点。需要说明的是,文中方法不仅可以集成 Openstack 和 Zabbix 的接口,只要是 API 格式不统一的系统,都可以使用该方法进行集成。

4 API 集成系统的设计与实现

4.1 在 Kong 中创建需要转发的 API

Kong 提供了完善的管理接口,用来完成对 Kong 的管理功能。默认情况下,Kong 服务的 8001 端口用于管理 API 的响应;8000 端口用于接收需要转发的请求。下面就是用 Kong 的管理接口来新建 Openstack 和 Zabbix 的转发 API。

创建 Openstack 的转发 API 的形式如下:

POST:http://KONG_HOST:8001/apis/

Body:{"name": "openstack_api", "uris": "openstack","upstream_url":"http://OPENSTACK_HOST","strip_uri":"true"}

这样创建 Openstack 转发 API 后,用户请求 http://KONG_HOST:8000/openstack,Kong 就能够将请求自动转发到 http://OPENSTACK_HOST。其中 KONG_HOST 和 OPENSTACK_HOST 分别表示 Kong 和 Openstack 的请求地址。

该请求 Body 中的参数含义如下:name 为转发 API 的名称;uris 为用户情况 Kong 的 API 路径;upstream_url 为请求转发的路径;strip_uri 为是否在转发过程中隐藏 uris 中设置的路径。

创建 Zabbix 的转发 API 的形式如下:

POST:http://KONG_HOST:8001/apis/

Body:{"name":"zabbix_api","uris":"zabbix", "upstream_url":"http://ZABBIX_HOST", "strip_uri":"true"}

Openstack 和 Zabbix 转发 API 创建成功后,Kong 可以根据用户请求的路径不同,将用户的请求分别转发给 Openstack 和 Zabbix。

4.2 Zabbix API 转发处理插件的设计并实现

上一节创建了用于转发的 API。但是 Zabbix 能够接收的请求并不是常用的 restfulAPI 请求。Zabbix 接收的请求方法、URL、Headers 和 Body 体形式如下:

POST http://zabbix /zabbix/api_jsonrpc.php

Headers:Content-Type: application/json-rpc

Body:{"jsonrpc":"2.0","method": "apiinfo.version","id":1,"auth":null,"params":{}}

Openstack 接收的请求方法、URL、Headers 和 Body 体形式如下:

POST http://openstack.com/openstack/api

Headers:Content-Type: application/json; X-Auth-Token: TOKEN-ID

Body:{"task_id": null}

通过 Zabbix 和 Openstack 的接口调用方式可以看出,两个系统的 API 系统有很大的差别。主要体现在 Headers 和 Body 信息中。如果想要使用统一的方式来完成两个系统的 API 接口调用,可以对其中一个系统

的 API 系统进行改进。但是改进 API 系统的工作是非常复杂和繁重的,需要花费大量的人力,而且还不一定能够做到稳定。

使用 Kong 可以很好地解决这个问题。Kong 提供了插件(Plugin)的功能。Kong 的插件能够绑定到不同的转发 API 上,并且能够对用户的请求和服务的响应做修改。通过 Kong 的插件,可以修改请求和响应的 url、headers 和 body。所以,在该系统中主要设计了针对修改 Zabbix 请求的 Kong 插件,并且绑定在 Kong 转发 Zabbix 的接口上,将 Zabbix 的接口调用方式改为 Openstack 的接口调用方式,实现了两个系统 API 请求的统一。

通过比较 Openstack 和 Zabbix 的接口调用方式可以看出,要想使用 Openstack 接口调用的方式调用 Zabbix 的接口,Kong 的插件所需要做的工作为:

(1)请求 Zabbix 接口时,在 Body 中添加“jsonrpc”和“id”参数。

(2)请求 Zabbix 接口时,读取 Header 中的 X-Auth-Token 数据,并写到 Body 的 auth 中。

(3)请求 Zabbix 接口时,将 Header 中的 Content-Type 改为 application/json-rpc。

(4)在请求返回的 body 中去除“jsonrpc”和“id”参数。

Kong 的插件是 lua[15] 语言书写,并且基于 lua-nginx-module 模块,所以必须熟悉 lua 语言、lua-nginx-module 模块和 Kong 的环境才能完成 Kong 插件的编写工作。

一个简单的 Kong 插件,必须包含两个文件:schema. lua 和 handler. lua。其中 schema. lua 文件是用于插件的用户配置,允许用户在使用插件时根据不同的需求做出不同的配置。下面 schema. lua 的内容设置了插件没有消费者:

```
return {no_consumer=true}
```

handler. lua 文件是插件的核心,所有的 API 处理流程和逻辑都在这个文件中完成。在该文件中,将一个完整的请求拆分为七个过程,并将七个过程使用七个函数来表示。在不同的 Kong 插件中可以重写这七个函数来完成不同阶段所执行的工作。这七个函数及其代表的执行阶段分别是:

(1)init_worker():初始化的工作阶段执行;

(2)certificate():SSL 握手的阶段执行;

(3)rewrite():重写阶段时执行;

(4)access():转发之前对请求处理的阶段执行;

(5)header_filter():接收到所有的响应头时执行;

(6)body_filter():接收到所有的响应 body 时执行;

(7)log():将响应信息发送给请求端之后执行。

因此,需要在 access()函数中完成添加“jsonrpc”和“id”参数的功能;在 access()函数中完成修改 Header 的功能;在 body_filter()函数中完成去掉“jsonrpc”和“id”参数的功能。

在 access()函数中完成读取 Header 的“X-Auth-Token”参数,“X-Auth-Token”、“jsonrpc”和“id”参数的功能代码如下:

```
localcjson=require "cjson"
localngx=ngx
localget_headers=ngx. req. get_headers
localset_headers=ngx. req. set_headers
set_header("Content-Type","application/json-rpc")
token_id=get_headers()["X-Auth-Token "]
ngx. req. read_body()
body=cjson. decode(ngx. req. get_body_data())
iftoken_id! =nil then
body["auth"]=token_id
end
body["jsonrpc"]="2. 0"
body["id"]=1
ngx. req. set_body_data(cjson. encode(body))
```

以上这段代码通过 lua-nginx-module 提供的功能,将 X-Auth-Token 的数据从 Header 中取出,并且填入 Body 中;然后在 Body 中添加 jsonrpc 和 id 的参数。

在 body_filter()函数中完成去掉响应 body 中的“jsonrpc”和“id”参数,其功能代码为:

```
local chunk, eof=ngx. arg[1], ngx. arg[2]
ifeof then
body=ngx. ctx. buffer
fori,v in ipairs(body) do
if table[i]=="jsonrpc" then
table. remove(body, i)
end
end
ngx. arg[1]=body
else
ngx. ctx. buffer=ngx. ctx. buffer. . chunk
ngx. arg[1]=nil
end
```

在 Kong 中,body 可能很大,需要多次接收才能全部接收完。所以以上代码块就是循环接收 body 的分块数据,组装成完整的 body,并且将 body 中的“jsonrpc”和“id”参数删除,最后返回到请求端。

通过 handler. lua 中的两个函数,就能够实现将 Zabbix 的接口调用形式与 Openstack 的接口调用形式统一。下面需要使插件生效,并且绑定到 Zabbix 的转发 API 上,使其处理 Zabbix 的请求。

4.3 插件发布并生效

发布插件非常简单,只需要将插件的文件存放在./kong/plugins 下,并在 Kong 的配置文件中增加 custom_plungins 字段,添加插件的名称,具体格式如下:

custom_plugins=zabbix-api-plugins

配置文件修改完成后,重启 Kong,插件发布完成。

将 zabbix-api-plugins 插件绑定到 4.1 节中创建的 Zabbix 的转发 API 上,使其生效。绑定的操作为:

POST:http://KONG_HOST:8001/apis/ zabbix_api/zabbix-api-plugins/

Body:{"name":"zabbix-api-plugins"}

将插件绑定的 Zabbix 转发到 API 上后,Kong 在转发请求 Zabbix 的 API 时,就能够自动完成 4.2 节中设置的格式转换。

5 结束语

通过 curl[16]命令来进行测试,验证该系统能够根据 Openstack 的格式来正常请求 Zabbix 的 API,测试结果如图 3 所示。

```
root@admin:~# curl -i -X POST \
>   --url http://localhost:8000/zabbix/ \
>   --data '{"method": "user.login",\
>            "params": {"user": "Admin",\
>                       "password": "zabbix"}}'
>   --header '{"Content-Type": "application/json"}'

{
    "result": "0424bd59b807674191e7d77572075f33"
}

root@admin:~#
```

图 3 测试结果

图 3 是通过 curl 对 Zabbix 登录接口的调用。在这次调用过程中,没有使用 Zabbix 的原始调用方式,但是能够正常返回调用结果,实现了 Zabbix 与 Openstack 的接口调用方式的统一。

以上介绍了基于接口网关 Kong 的云计算平台 API 集成系统的设计与实现。通过 API 集成系统转发所有云计算平台子系统的 API 请求,实现了地址的隐藏和统一调用的功能。而且通过给不同的转发 API 绑定不同的插件,可以完成使用统一的格式来调用不同的子系统 API。减少了开发人员和接口使用人员的工作量,也减少了二次开发人员的学习成本。而且 Kong 支持多节点的扩展,能够保证多并发下的响应速度。

因此,通过该系统在不损失响应速率的基础上,解决了不同系统 API 的集成,也实现了调用 API 接口的格式统一。

参考文献:

[1] 武志学. 云计算虚拟化技术的发展与趋势[J]. 计算机应用,2017,37(4):915-923.

[2] 肖伟民,邓浩江,胡琳琳,等. 基于容器的 Web 运行环境轻量级虚拟化方法[J]. 计算机应用研究,2018,35(6):1768-1772.

[3] 杨 飞,朱志祥,梁小江. 基于 Ceph 对象存储集群的高可用设计与实现[J]. 微电子学与计算机,2016,33(1):60-64.

[4] JIANG Liqiu. Comparing function of OpenStack and VMware[C]//Proceedings of 2017 2nd international conference on wireless communication and network engineering. [s. l.]:Science and Engineering Research Center,2017.

[5] 郑 萌. 基于 Zabbix 的云监控系统的设计与实现[D]. 成都:电子科技大学,2017.

[6] 徐 震,焦文彬. RabbitMQ 小消息确认机制优化[J]. 计算机系统应用,2018,27(3):252-257.

[7] 徐 敏,李 明,郑建忠,等. 基于 OpenStack 的 Swift 负载均衡算法[J]. 计算机系统应用,2018,27(1):127-131.

[8] 刘 卓,张向利. 基于 Nginx 的负载均衡集群设计与实现[J]. 桂林电子科技大学学报,2017,37(6):490-493.

[9] KALIDS. Big-data NoSQL databases:a comparison and analysis of "big-table","dynamo DB", and "Cassandra"[C]//Proceedings of 2017 IEEE 2nd international conference on big data analysis. Beijing:IEEE,2017.

[10] 宗 平,李 雷. PostgreSQL 与 MongoDB 处理非结构化数据性能比较[J]. 计算机工程与应用,2017,53(7):104-108.

[11] 柳 静,邵 华. 云计算虚拟化集群技术研究[J]. 电脑知识与技术,2016,12(23):219-220.

[12] 孙伟龙. 基于 IaaS 云计算的 Web 应用技术研究[D]. 南京:南京理工大学,2011.

[13] 白忠军,孔广黔,吴 云. 基于 RESTful 的校园二手商品交易系统的设计与实现[J]. 计算技术与自动化,2018,37(1):126-130.

[14] 柯 兢. 基于 JSON-RPC 协议的可穿戴设备跨平台开发[D]. 广州:华南理工大学,2016.

[15] 林 巧,杨坚坚. Lua 语言在轻量级 Web 服务器设计中的应用[J]. 计算机系统应用,2016,25(7):124-129.

[16] 王冠一. 基于 Struts 和 Curl 的测试信息管理系统的设计与实现[D]. 成都:电子科技大学,2013.