玩转 Nginx 之：使用 Lua 扩展 Nginx 功能 原

大数据之路 发布于 2015/08/19 05:02 字数 5614 阅读 7969 收藏 39 点赞 2 评论 0

Nginx Tengine Lua LuaJIT

1、Nginx 简介

Nginx 作为一款面向性能设计的HTTP服务器，相较于Apache、lighttpd具有占有内存少，稳定性高等优势。其流行度越来越高，应用也越来越广泛，常见的应用有：网页服务器、反向代理服务器以及电

子邮件（IMAP/POP3）代理服务器，高并发大流量站点常用来做接入层的负载均衡，还有非常常见的用法是作为日志采集服务器等。

Nginx 整体采用模块化设计，有丰富的模块库和第三方模块库，配置灵活。其中模块化设计是nginx的一大卖点，甚至http服务器核心功能也是一个模块。要注意的是：nginx的模块是静态的，添加和删除模块都要对nginx进行重新编译，这一点与Apache的动态模块完全不同。不过后来淘宝做了二次开发开源的 tengine 是支持 官方所有的 HTTP 模块动态加载而不必重新编译 Nginx，除非是第三方模块才需要重新编译。因此，在生产环境中，推荐用淘宝开源的 tengine，本文也以 tengine 作为示例。

虽然 Nginx 有如此强大的性能以及众多的三方模块支持，但每次重新编译以及寻找三方模块对生产环境来说还是不可接受的，幸运的是，Nginx 它是支持客户自己 Lua 脚本编程扩展相应的功能的，而且可以热加载，这就给生产环境带来了无限可能。比如我现在想要直接用Nginx + redis 做反爬虫和频率限制，Nginx + Kafka 做日志的实时流处理等等。

注：lvs 和 nginx 的负载均衡区别：

LVS：Linux Virtual Server，基于IP的负载均衡和反向代理技术，所以它几乎可以对所有应用做负载均衡，包括http、数据库、在线聊天室等等，LVS工作在4层，在Linux内核中作四层交换，只花128个字节记录一个连接信息，不涉及到文件句柄操作，故没有65535最大文件句柄数的限制。LVS性能很高，可以支持100～400万条并发连接。抗负载能力强、是工作在网络4层之上仅作分发之用，**没有流量的产生**，这个特点也决定了它在负载均衡软件里的性能最强的，对内存和cpu、IO资源消耗比较低。

Nginx：基于HTTP的负载均衡和反向代理服务器，Nginx工作在网络的7层，所以它可以针对http应用本身来做分流策略，比如针对域名、URL、目录结构等，相比之下LVS并不具备这样的功能，能够很好地支持虚拟主机，可配置性很强，大约能支持3～5万条并发连接。

2、Lua 简介

Lua 是一个简洁、轻量、可扩展的脚本语言，也是号称性能最高的脚本语言，用在很多需要性能的地方，比如：游戏脚本，nginx，wireshark的脚本，当你把他的源码下下来编译后，你会发现解释器居然不到200k，非常变态。。。很多应用程序使用Lua作为自己的嵌入式脚本语言，以此来实现可配置性、可扩展性。

Lua原生支持的数据类型非常之少，它只提供了nil、数字（缺省是双精度浮点数，可配置）、布尔量、字符串、表、子程序、协程（coroutine）以及用户自定义数据这8种。但是其处理表和字符串的效率非常之高，加上元表的支持，开发者可以高效的模拟出需要的复杂数据类型（比如集合、数组

等）。Lua是一个动态弱类型语言，支持增量式垃圾收集策略。有内建的，与操作系统无关的协作式多线程（coroutine）支持。它还可以用于嵌入式硬件，不仅可以嵌入其他编程语言，而且可以嵌入微处理器中。

3、nginx执行步骤

nginx在处理每一个用户请求时，都是按照若干个不同的阶段依次处理的，与配置文件上的顺序没有关系，详细内容可以阅读《深入理解nginx:模块开发与架构解析》这本书，这里只做简单介绍。nginx实际把http请求处理流程划分为了11个阶段，这样划分的原因是将请求的执行逻辑细分，以模块为单位进行处理，各个阶段可以包含任意多个HTTP模块并以流水线的方式处理请求。这样做的好处是使处理过程更加灵活、降低耦合度。这11个HTTP阶段如下所示：

1）NGX_HTTP_POST_READ_PHASE：

接收到完整的HTTP头部后处理的阶段，它位于uri重写之前，实际上很少有模块会注册在该阶段，默认的情况下，该阶段被跳过。

2）NGX_HTTP_SERVER_REWRITE_PHASE：

URI与location匹配前，修改URI的阶段，用于重定向，也就是该阶段执行处于server块内，location块外的重写指令，在读取请求头的过程中nginx会根据host及端口找到对应的虚拟主机配置。

3）NGX_HTTP_FIND_CONFIG_PHASE：

根据URI寻找匹配的location块配置项阶段，该阶段使用重写之后的uri来查找对应的location，值得注意的是该阶段可能会被执行多次，因为也可能有location级别的重写指令。

4）NGX_HTTP_REWRITE_PHASE：

上一阶段找到location块后再修改URI，location级别的uri重写阶段，该阶段执行location基本的重写指令，也可能会被执行多次。

5）NGX_HTTP_POST_REWRITE_PHASE：

防止重写URL后导致的死循环，location级别重写的后一阶段，用来检查上阶段是否有uri重写，并根据结果跳转到合适的阶段。

6）NGX_HTTP_PREACCESS_PHASE：

下一阶段之前的准备，访问权限控制的前一阶段，该阶段在权限控制阶段之前，一般也用于访问控制，比如限制访问频率，链接数等。

7）NGX_HTTP_ACCESS_PHASE：

让HTTP模块判断是否允许这个请求进入Nginx服务器，访问权限控制阶段，比如基于ip黑白名单的权限控制，基于用户名密码的权限控制等。

8）NGX_HTTP_POST_ACCESS_PHASE：

访问权限控制的后一阶段，该阶段根据权限控制阶段的执行结果进行相应处理，向用户发送拒绝服务的错误码，用来响应上一阶段的拒绝。

9）NGX_HTTP_TRY_FILES_PHASE：

为访问静态文件资源而设置，try_files指令的处理阶段，如果没有配置try_files指令，则该阶段被跳过。

10）NGX_HTTP_CONTENT_PHASE：

处理HTTP请求内容的阶段，大部分HTTP模块介入这个阶段，内容生成阶段，该阶段产生响应，并发送到客户端。

11）NGX_HTTP_LOG_PHASE：

处理完请求后的日志记录阶段，该阶段记录访问日志。

以上11个阶段中，HTTP无法介入的阶段有4个：

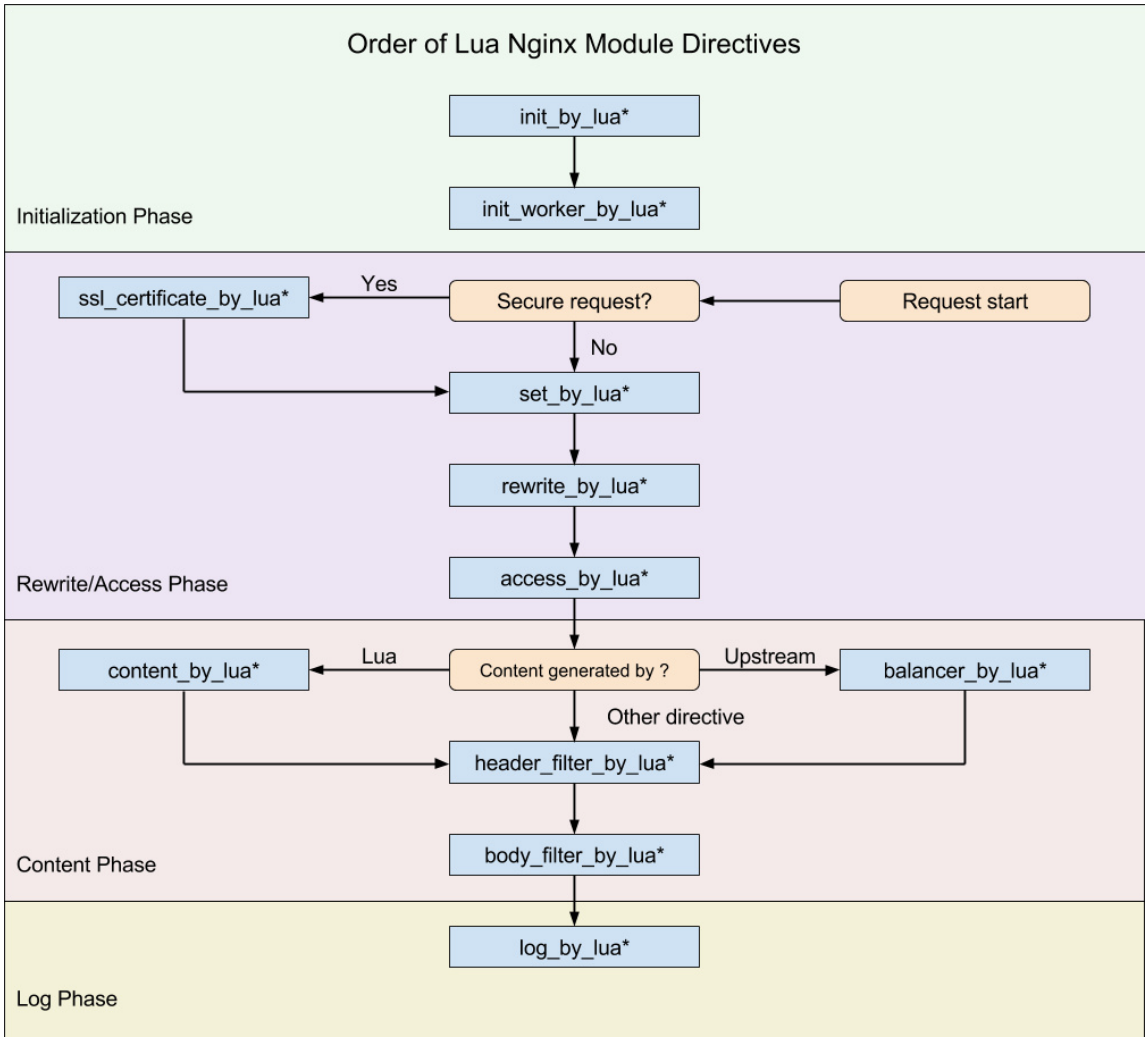
3）NGX_HTTP_FIND_CONFIG_PHASE

5）NGX_HTTP_POST_REWRITE_PHASE

8）NGX_HTTP_POST_ACCESS_PHASE

9）NGX_HTTP_TRY_FILES_PHASE

剩余的7个阶段，HTTP模块均能介入，每个阶段可介入模块的个数也是没有限制的，多个HTTP模块可同时介入同一阶段并作用于同一请求。



图：Nginx 模块执行顺序与阶段

Refer：

Nginx请求处理流程你了解吗?

https://mp.weixin.qq.com/s/otQIhuLABU3omOLtRfJnZQ

4、ngx_lua 运行指令

ngx_lua属于nginx的一部分，它的执行指令都包含在nginx的11个步骤之中了，相应的处理阶段可以做插入式处理，即可插拔式架构，不过ngx_lua并不是所有阶段都会运行的；另外指令可以在http、server、server if、location、location if几个范围进行配置：

指令	所处处理阶段	使用范围	解释

init_by_lua init_by_lua_file	loading-config	http	nginx Master进程加载配置时执行； 通常用于初始化全局配置/预加载Lua模块
init_worker_by_lua init_worker_by_lua_file	starting-worker	http	每个Nginx Worker进程启动时调用的计时器，如果Master进程不允许则只会在init_by_lua之后调用； 通常用于定时拉取配置/数据，或者后端服务的健康检查
set_by_lua set_by_lua_file	rewrite	server,server if,location,location if	设置nginx变量，可以实现复杂的赋值逻辑；此处是阻塞的，Lua代码要做到非常快；
rewrite_by_lua rewrite_by_lua_file	rewrite tail	http,server,location, location if	rrewrite阶段处理，可以实现复杂的转发/重定向逻辑；
access_by_lua access_by_lua_file	access tail	http,server,location, location if	请求访问阶段处理，用于访问控制
content_by_lua content_by_lua_file	content	location，location if	内容处理器，接收请求处理并输出响应
header_filter_	output-	http，server，	设置header和cookie

by_lua header_filter_by_lua_file	header-filter	location，location if	
body_filter_by_lua body_filter_by_lua_file	output-body-filter	http，server，location，location if	对响应数据进行过滤，比如截断、替换。
log_by_lua log_by_lua_file	log	http，server，location，location if	log阶段处理，比如记录访问量/统计平均响应时间

关于这部分详细可以参考这篇：

Refer [4] nginx与lua的执行顺序和步骤说明

Refer [5] ngx_lua用例说明

5、安装 tengine 以及 Lua 扩展

（1）先安装Nginx需要的一些类库：

yum install gcc

yum install gcc-c++

注：此步骤只是在你的系统没有安装 gcc/gcc-c++ 的情况下才需要自行编译安装。

（2）编译安装库LuaJit-2.0.3：

./configure --prefix=/usr/local/luajit

make PREFIX=/usr/local/luajit

make install PREFIX=/usr/local/luajit