Nginx初探

0、基本资料：

官网：http://nginx.org/

源码：https://github.com/nginx/nginx.git

语音：C

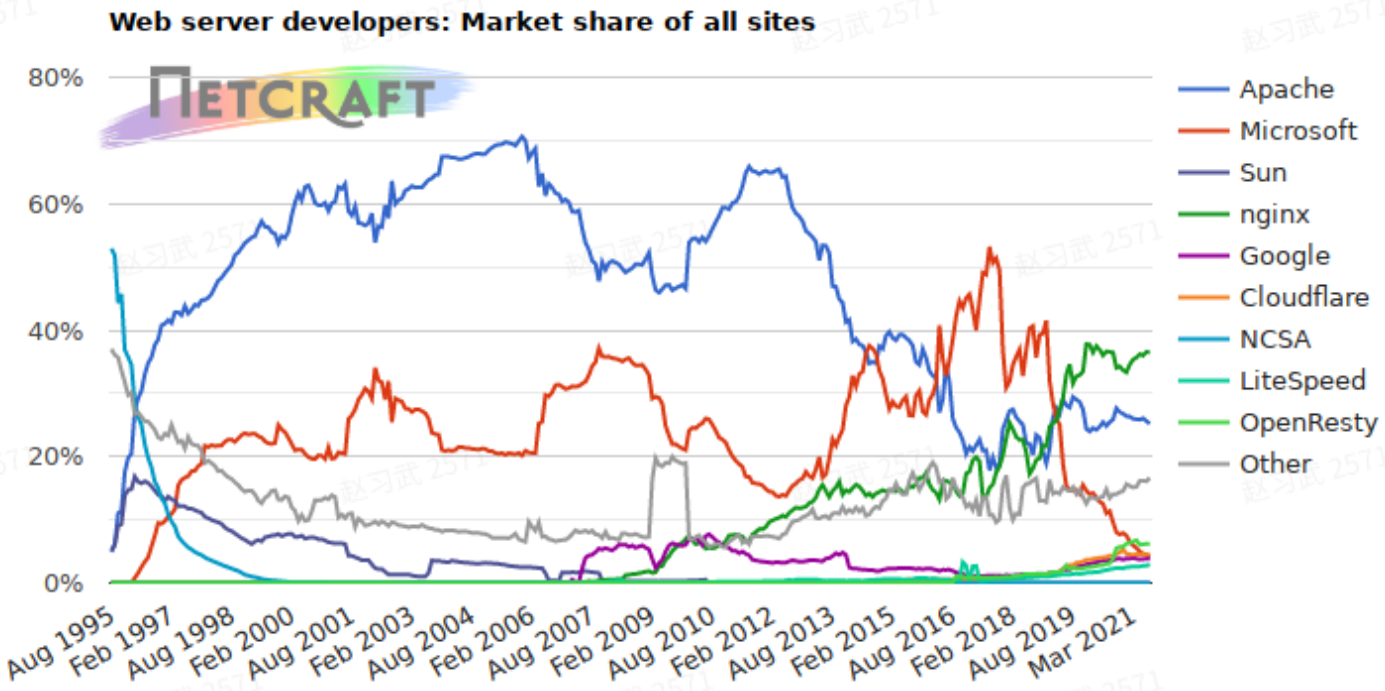
模式：多进程

io模型：异步非阻塞

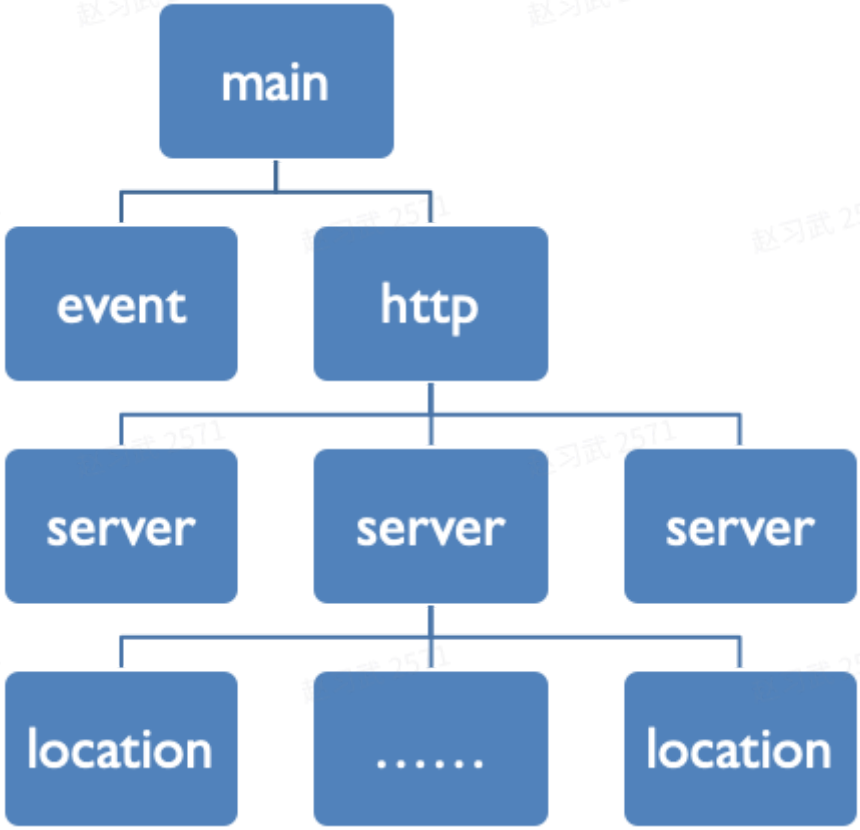
进程间通信：信号量，共享内存

忌讳：阻塞的系统调用

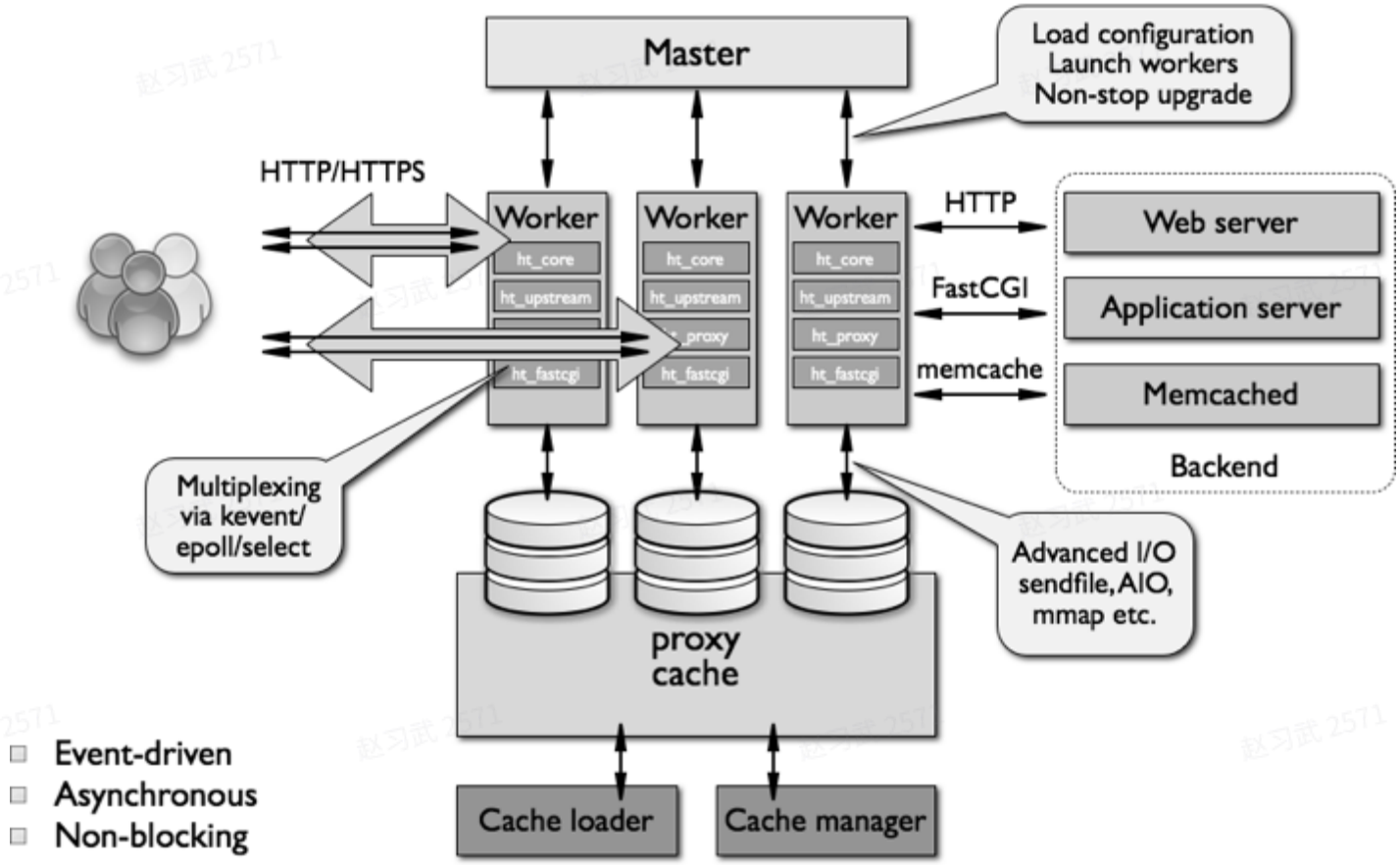
市场占有率：36.48%



1、配置结构：


main
event
http
server
server
server
location
……
location

2、整体架构


Master
Load configuration
Launch workers
Non-stop upgrade
HTTP/HTTPS
Worker
Worker
Worker
ht_core
ht_upstream
ht_proxy
ht_fastcgi
HTTP
FastCGI
memcache
Web server
Application server
Memcached
Backend
Multiplexing via kevent/ epoll/select
Advanced I/O sendfile, AIO, mmap etc.
proxy cache
Cache loader
Cache manager
Event-driven
Asynchronous
Non-blocking

3、源码结构

目录结构：

```
zhaoxiwu@Mac in ~/code/nginx at (759af1a4...)
> ll
total 1840
drwxr-xr-x  25 zhaoxiwu  staff     800 Jun 29  2020 auto
drwxr-xr-x  11 zhaoxiwu  staff     352 Mar 15  2018 conf
drwxr-xr-x   6 zhaoxiwu  staff     192 Mar 15  2018 contrib
drwxr-xr-x  10 zhaoxiwu  staff     320 Mar 15  2018 docs
drwxr-xr-x   4 zhaoxiwu  staff     128 Mar 15  2018 misc
drwxr-xr-x   9 zhaoxiwu  staff     288 May 27  2020 src
-rw-------  1 zhaoxiwu  staff  938145 Mar 15  2018 tags
zhaoxiwu@Mac in ~/code/nginx at (759af1a4...)
> cd src/
zhaoxiwu@Mac in ~/code/nginx/src at (759af1a4...)
> ll
total 0
drwxr-xr-x  69 zhaoxiwu  staff  2208 Feb  3  2021 core
drwxr-xr-x  19 zhaoxiwu  staff   608 May 27  2020 event
drwxr-xr-x  33 zhaoxiwu  staff  1056 Jun  4  2020 http
drwxr-xr-x  20 zhaoxiwu  staff   640 Mar 15  2018 mail
drwxr-xr-x   4 zhaoxiwu  staff   128 Mar 15  2018 misc
drwxr-xr-x   6 zhaoxiwu  staff   192 Mar 15  2018 mysql
drwxr-xr-x   4 zhaoxiwu  staff   128 Mar 15  2018 os
zhaoxiwu@Mac in ~/code/nginx/src at (759af1a4...)
```

.

```
├──── auto        自动检测系统环境以及编译相关的脚本
|   ├──── cc      关于编译器相关的编译选项的检测脚本
|   ├──── lib     nginx编译所需要的一些库的检测脚本
|   ├──── os      与平台相关的一些系统参数与系统调用相关的检测
|   └──── types    与数据类型相关的一些辅助脚本
├──── conf        存放默认配置文件，在make install后，会拷贝到安装目录中去
├──── contrib      存放一些实用工具，如geo配置生成工具（geo2nginx.pl）
├──── html        存放默认的网页文件，在make install后，会拷贝到安装目录中去
├──── man         nginx的man手册
```

└──── src　　存放nginx的源代码

├──── core　　nginx的核心源代码，包括常用数据结构的定义，以及nginx初始化运行的核心代码如main函数

├──── event　　对系统事件处理机制的封装，以及定时器的实现相关代码

│　└──── modules 不同事件处理方式的模块化，如select、poll、epoll、kqueue等

├──── http　　nginx作为http服务器相关的代码

│　└──── modules 包含http的各种功能模块

├──── mail　　nginx作为邮件代理服务器相关的代码

├──── misc　　一些辅助代码，测试c++头的兼容性，以及对google_perftools的支持

└──── os　　主要是对各种不同体系统结构所提供的系统函数的封装，对外提供统一的系统调用接

源码阅读：

nginx编程语言是c，工程量也比较大尽量用一些IDE来阅读源码，更方便查看函数调用关系。如果用vim可以用ctags -R 来生成函数调用关系，也可以实现各种跳转。

具体方法: 在src 目录下执行 ctags -R， 当前目录会升个一个tags文件，然后再通目录打开想看的文件即可，当需要查看引用的函数具体实现时同时按下 ctrl+] ，回退用ctrl+t

模块划分：

Event module： 事件处理机制框架，例如timer，io多路复用等

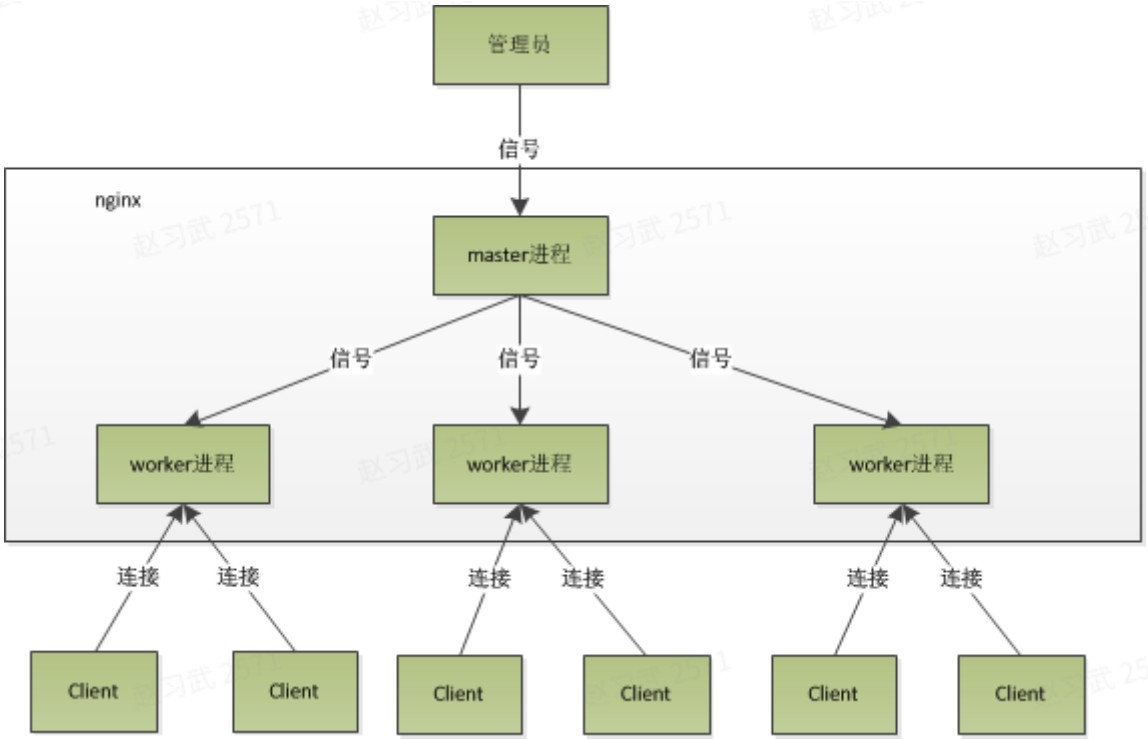
Phase handler：nginx处理客户端请求的地方，例如频率限制，黑白名单，http body处理等，也是开发者应用做多的。此次会把request拆分成11个阶段，类似工厂流水线生成商品，content阶段只能同时使用一个同类型handler

Output filter：nginx封装处理response的模块，一般会循环执行完所有filter才会结束

Upstream：nginx作为反向代理与上游服务交互的模块，也是handler模式，基本写法同Phase handler，但略有差异，开发者也经常出没在这个模块定制一些自己的功能。

Load Balancer：配合Upstream模块一起工作，通过各种算法获取上游服务器地址，同upstream模块，很多开发者也会根据业务需求自己开发一些LoadBalance算法。

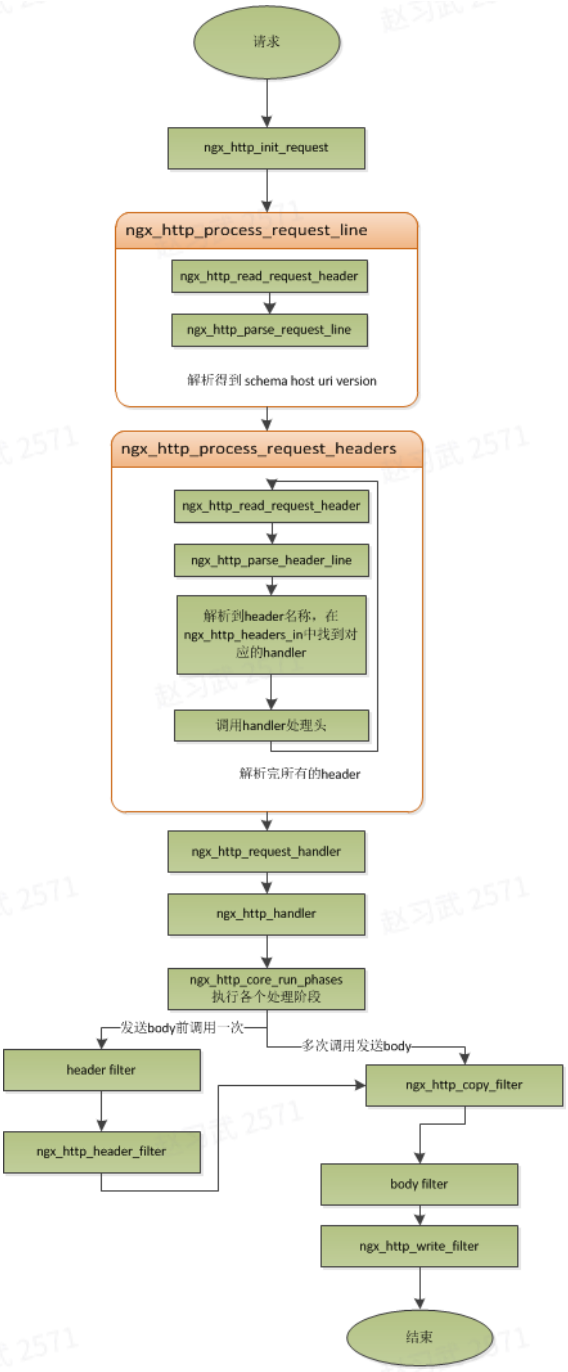
4、进程管理



5、Phase handler

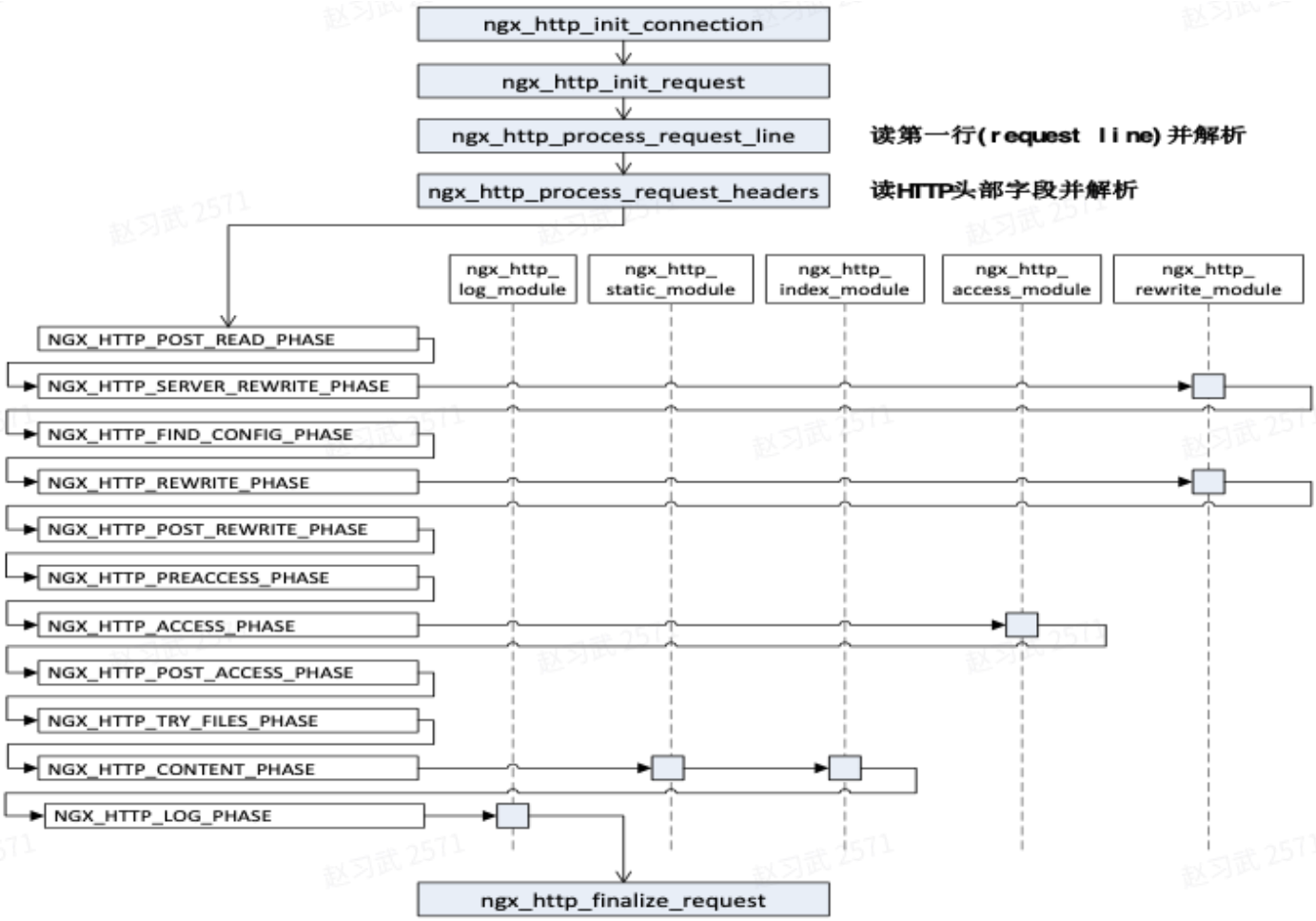
处理流程

1. 初始化HTTP Request（读取来自客户端的数据，生成HTTP Request对象）。
2. 处理请求行&请求头，调用与此请求（URL或者Location）关联的handler。
3. 处理请求体，依次调用各phase handler进行处理。



多阶段处理请求：

nginx实际把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，各阶段按照处理时机定义了清晰的执行语义，开发者可以很容易分辨自己需要开发的模块应该定义在什么阶段，下面介绍一下各阶段:



NGX_HTTP_POST_READ_PHASE: 接收完请求头之后的第一个阶段，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；

NGX_HTTP_SERVER_REWRITE_PHASE: server级别的uri重写阶段，也就是该阶段执行处于server块内，location块外的重写指令，前面的章节已经说明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；

NGX_HTTP_FIND_CONFIG_PHASE: 寻找location配置阶段，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；

NGX_HTTP_REWRITE_PHASE: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；

NGX_HTTP_POST_REWRITE_PHASE: 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； 该阶段不能注册handler

---

NGX_HTTP_PREACCESS_PHASE: 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；

NGX_HTTP_ACCESS_PHASE: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；该阶段支持同时注册多个handler

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；该阶段不能注册handler

---

NGX_HTTP_TRY_FILES_PHASE: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；该阶段也不能注册handler。

NGX_HTTP_CONTENT_PHASE:内容生成阶段，该阶段产生响应，并发送到客户端；只执行一个content_handler

--output filter --

NGX_HTTP_LOG_PHASE:日志记录阶段，该阶段记录访问日志。LOG阶段和其他阶段的不同点有两个，一是执行点是在ngx_http_free_request中，二是这个阶段的所有handler都会被执行。

6、Output Filter

header filter， body Filter

7、Upstream&LoadBalance


转发协议模块
create_request
均衡算法模块
http_proxy
reinit_request
upstream->peer.init
ip_hash
process_header
upstream