



火龙果·整理
uml.org.cn

从代码看Nginx运维本质

常见Web框架



火龙果 · 整理
uml.org.cn

可伸缩性、可用性

Nginx在负载均衡和Web加速服务器层面负有盛名



负载均衡
可伸缩性，容灾



WEB加速服务器
静态内容、高层次内容缓存



应用服务器
动态内容处理，如django、tomcat



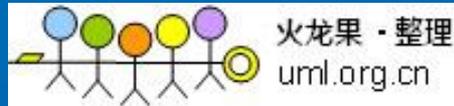
缓存
低层次内容缓存，如redis



数据库
数据持久化



为什么选择Nginx?



更快



节约机器、降低成本

单节点支撑更大的负载

每请求消耗更少的资源

投入产出比高

广泛使用，经过全面考验

扩展性高，大量模块提供了
足够丰富的功能

可以不造轮子，只搭积木

运维nginx的步骤



编译

准备环境
理解、执行configure
编译

01

修改配置

熟悉配置格式
了解如何明确用法
理解脚本式配置

03

error日志

编译时--with-debug打开
定位问题时配置日志级别
debug日志定位配置错误

05

部署

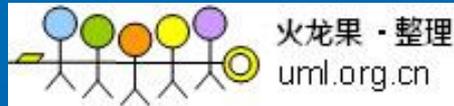
安装到正确位置
开机自启
Access日志分割

access日志

熟悉可利用的变量

```
goaccess -f access.log -o report.html --  
real-time-html --ws-url=zlldata.com
```

提升性能



提升传输效率

长连接代替短连接
Gzip压缩降低带宽

限制速度

保护应用服务器



缓存

有效的缓存往往是提升性能的杀器

减少进程消耗

降低写日志磁盘IO
使用sendfile

修改操作系统配置

更好的适合高并发场景

01 架构优化

缓存
keepalive长连接
gzip压缩

02 Nginx优化

限制流量
access日志批量输出
启用sendfile等高性能配置

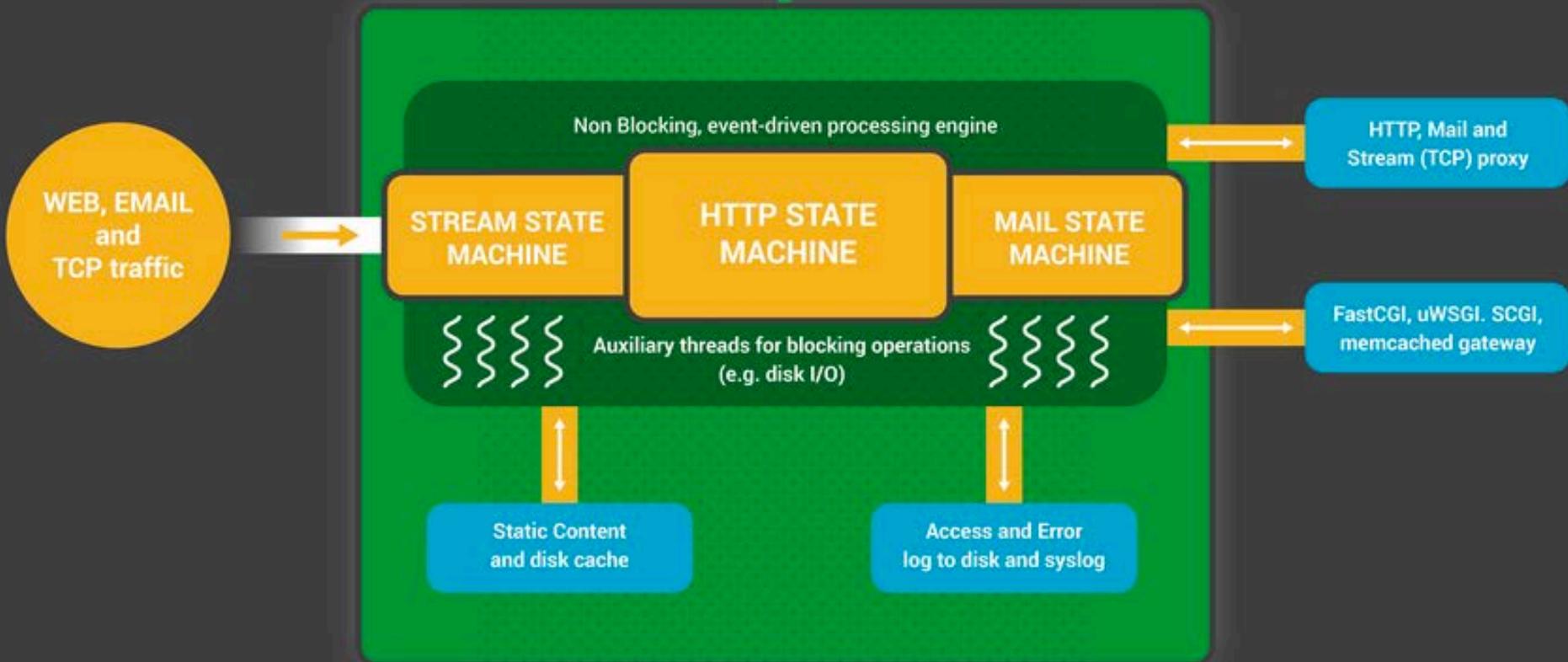
03 Linux优化

提高Backlog、文件描述符、端口范围限制
TCP连接传输优化

Nginx能做些什么？

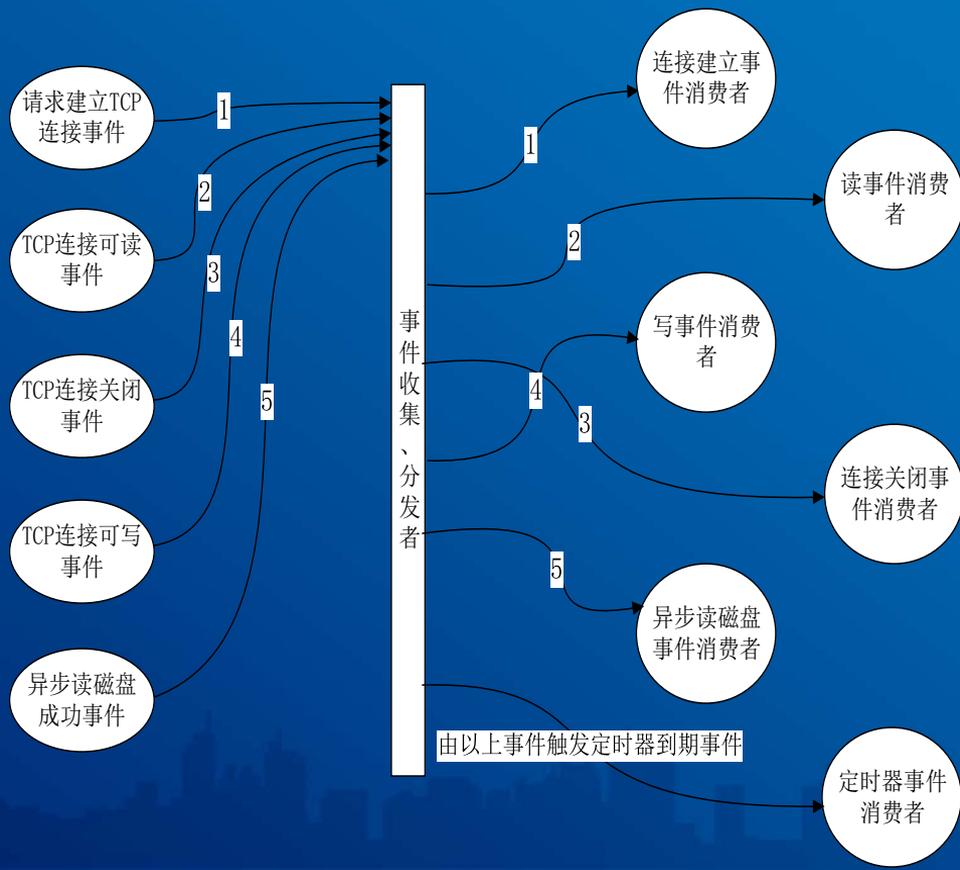
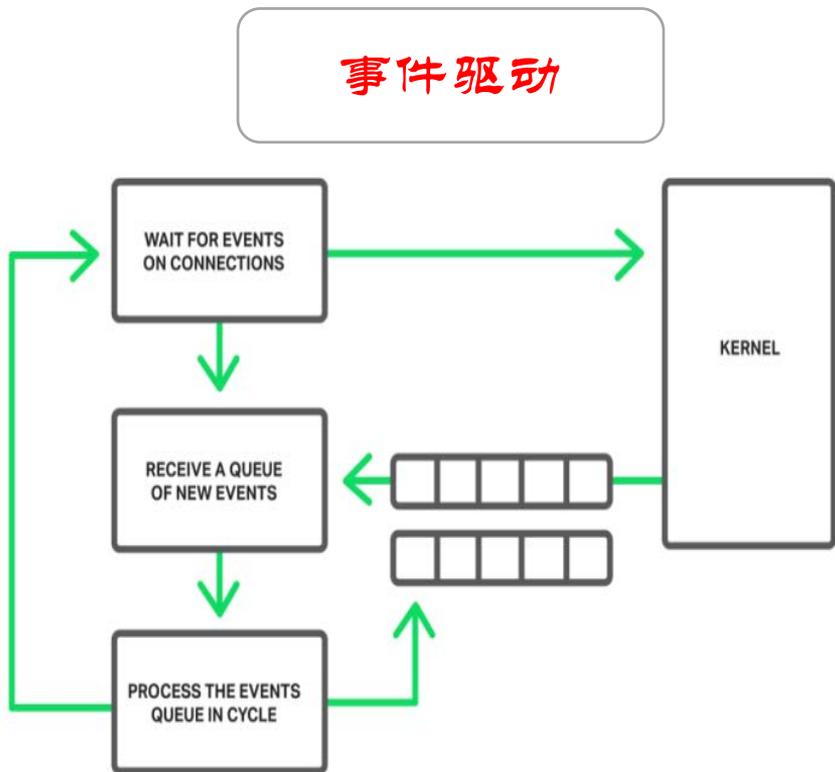
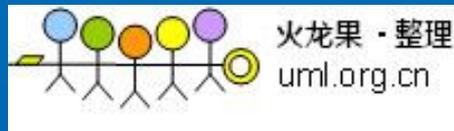


火龙果 · 整理
um1.org.cn



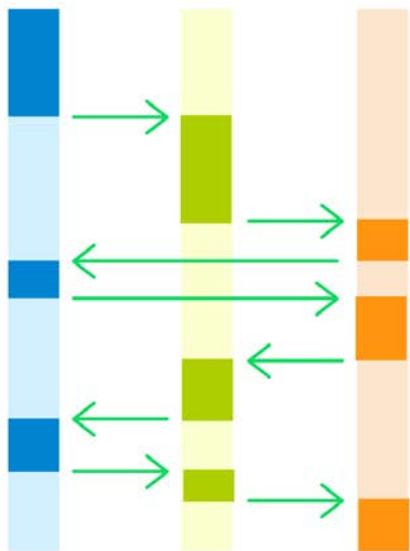
事件驱动框架

events{use epoll;}



TRADITIONAL SERVER

PROCESS 1 PROCESS 2 PROCESS 3



NGINX WORKER

PROCESS



TASK SWITCHES



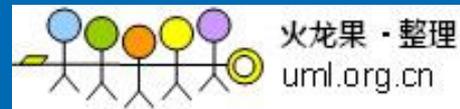
PROCESSING REQUEST 1



PROCESSING REQUEST 2



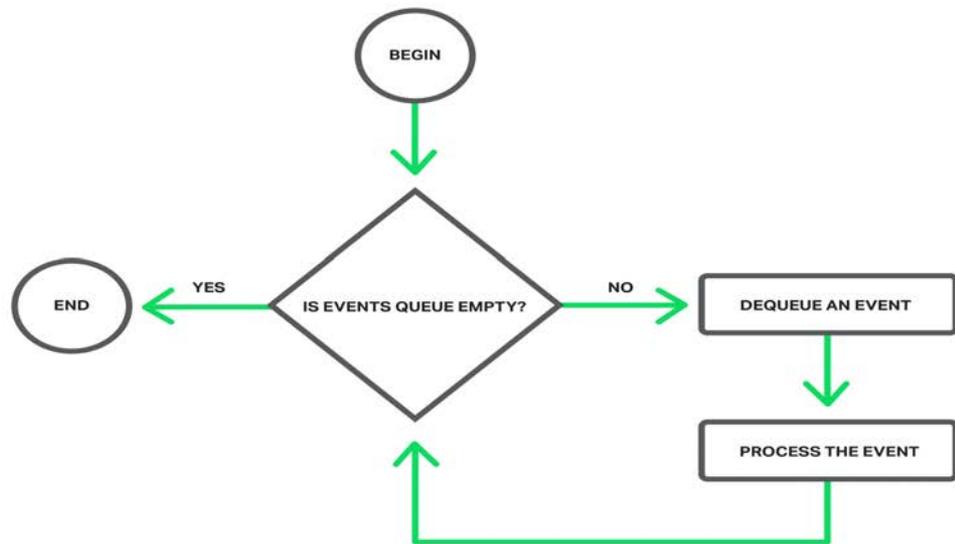
PROCESSING REQUEST 3



事件驱动框架
的优点：

很少的上下文
切换

THE EVENTS QUEUE PROCESSING CYCLE



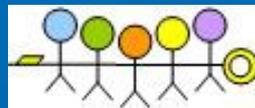
BLOCKING OPERATION



事件驱动框架的
缺点:

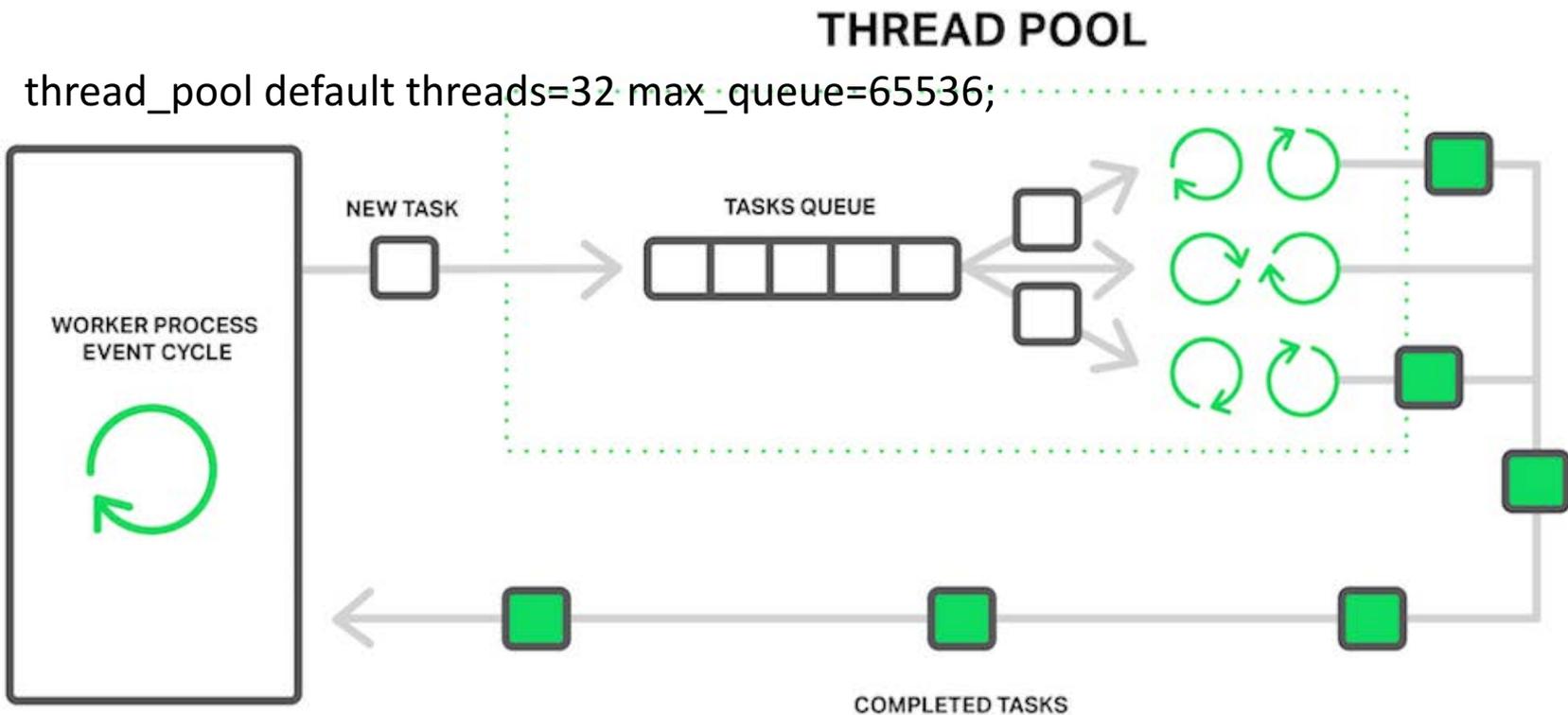
阻塞操作可以轻
易的造成性能下
降

线程池与事件驱动框架的结合



火龙果 · 整理
uml.org.cn

sendfile() on purpose only works on things that use the page cache. ---Linus Torvalds



Master-Worker进程结构



全局配置

```
master_process on;  
worker_processes auto;  
worker_cpu_affinity auto;  
  
pid nginx.pid;  
worker_priority -10;
```

- 清闲的master进程

启动、监控worker、cache进程
接收信号命令
读取、解析、重载配置文件
执行binary升级

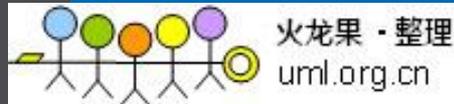
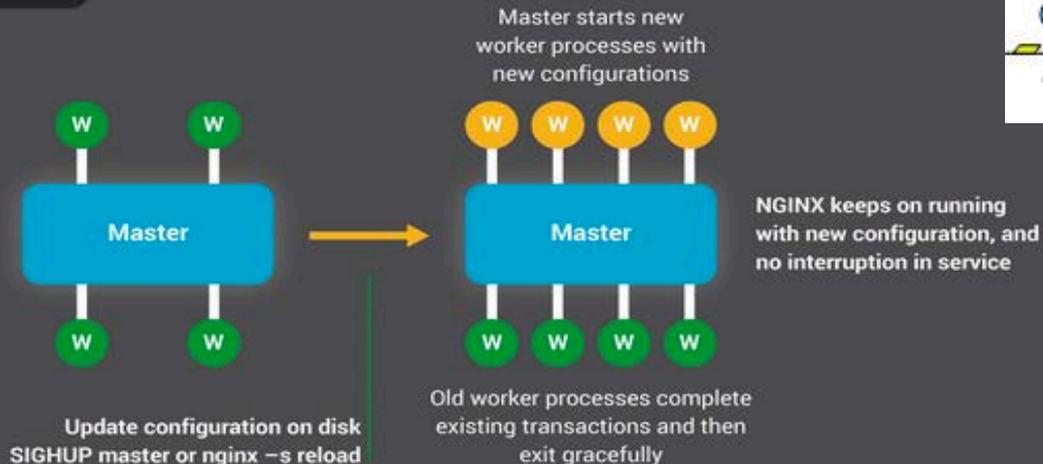
- 繁忙的worker进程

建立连接，处理各种请求，包括
http请求、mail请求、tcp传输

- Cache相关进程

处理文件缓存

Load new configuration with no downtime



Master-worker
架构的优点:

nginx -s reload

Load new NGINX binary with no downtime



升级Nginx

监听端口复用reuseport

master建立一个监听socket, worker进程间共享



KERNEL

SOCKET LISTENER

WORKER

WORKER

WORKER

WORKER

```
server {  
listen 80;  
}
```

master为每个worker进程建立独立的socket



KERNEL

SOCKET LISTENER

SOCKET LISTENER

SOCKET LISTENER

SOCKET LISTENER

WORKER

WORKER

WORKER

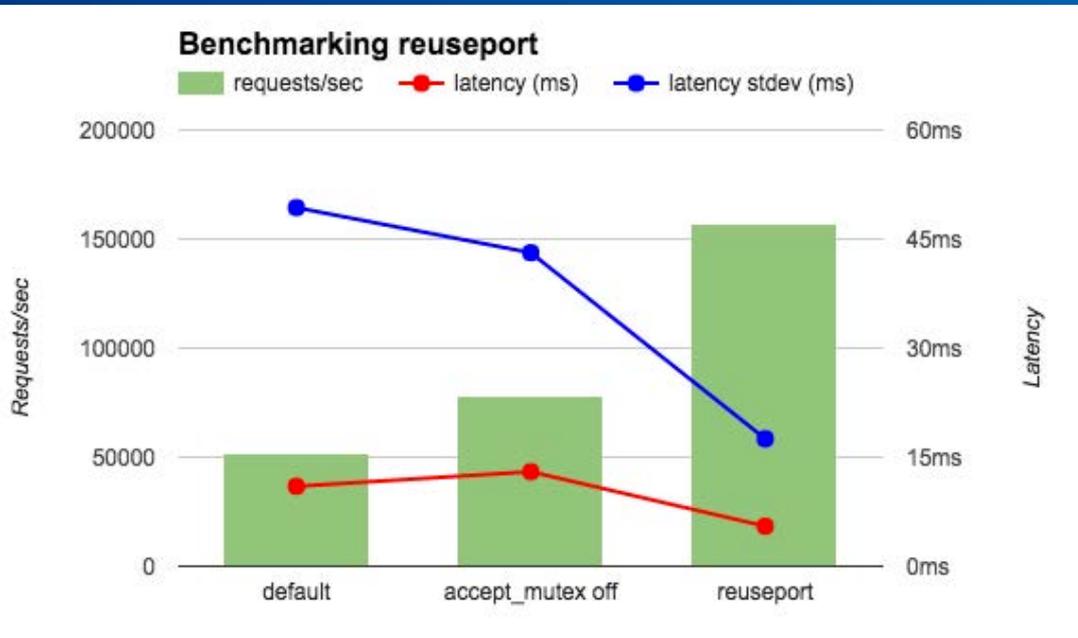
WORKER

```
server {  
listen 80  
reuseport;  
}
```

用ISO1可以观察



reuseport性能

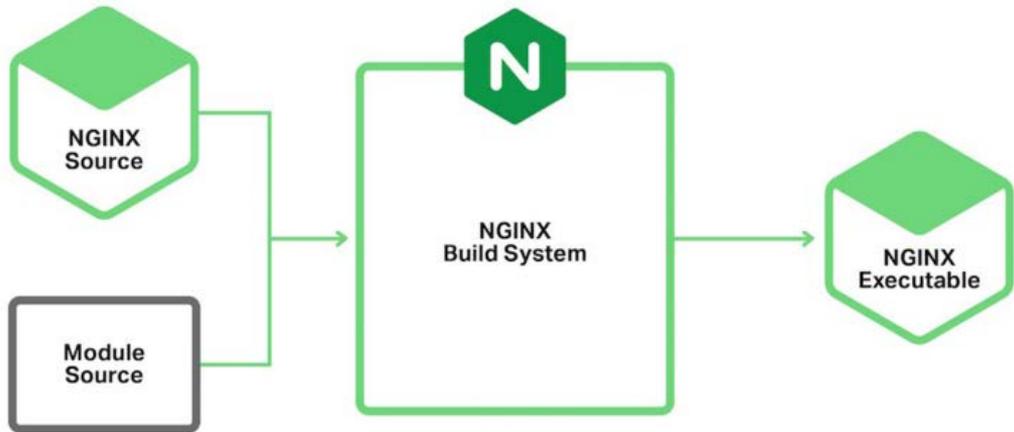


Linux3.9内核以上

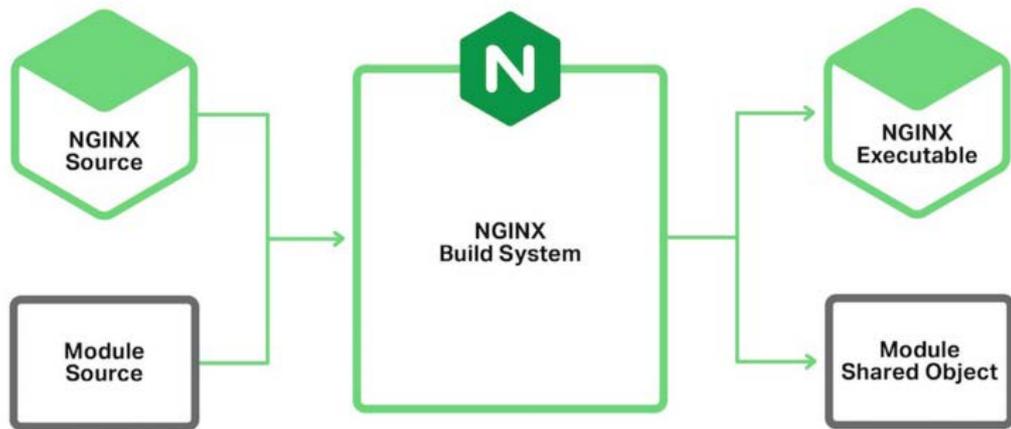
```
events  
{accept_mutex on;}
```

```
events  
{accept_mutex off;}
```

```
http {  
listen 80  
reuseport;}
```

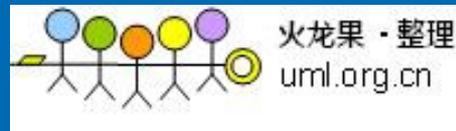



`load_module modules/nginx_mail_module.so`



Dynamic modules are compiled into a separate binary module shared object

动态模块



Configure加入模块到c代码

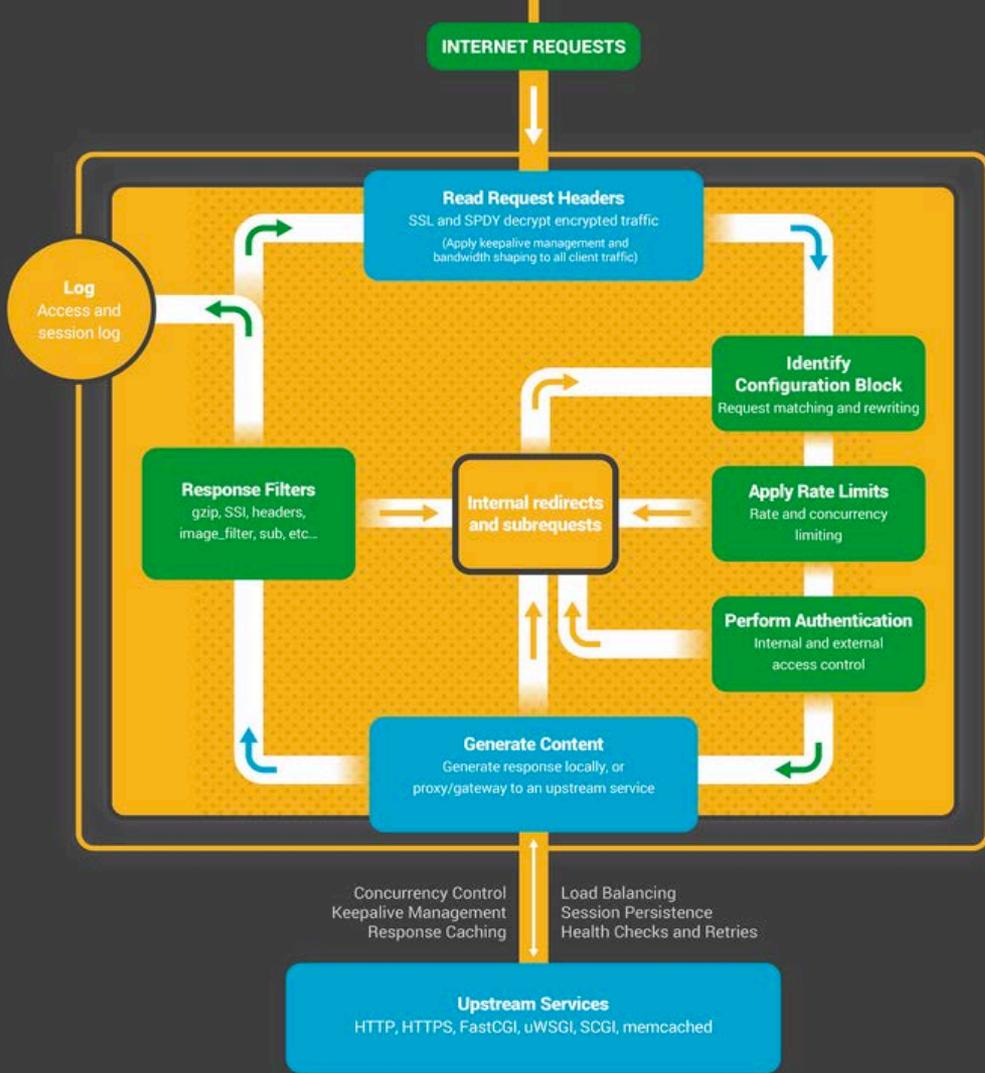
编译进binary

启动时初始化模块数组

读取load_module配置

打开动态库并加入模块数组

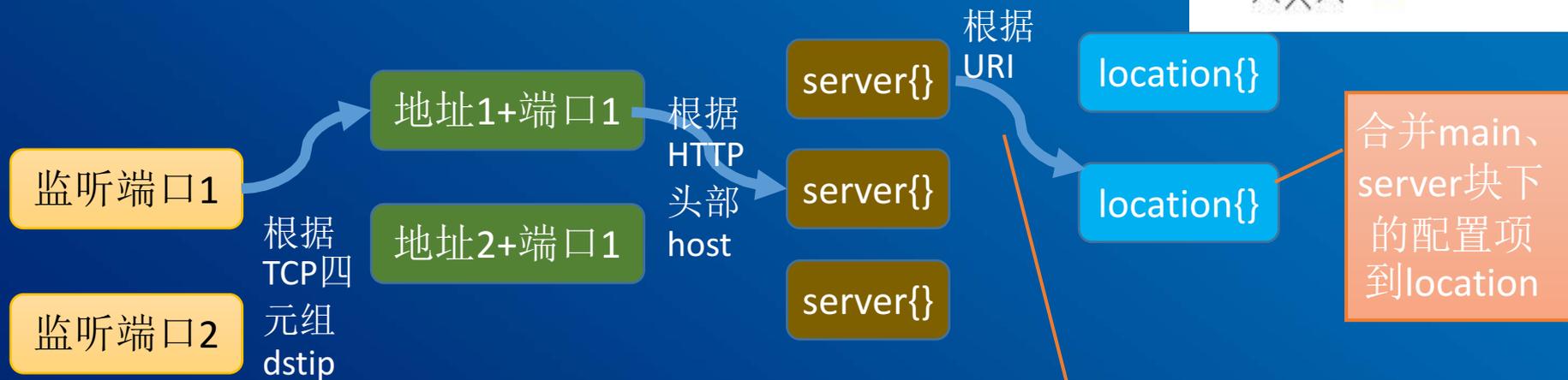
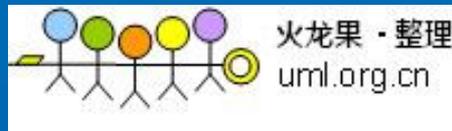
基于模块数组开始初始化



火龙果 · 整理
umml.org.cn

NGX_HTTP_POST_READ_PHASE	
NGX_HTTP_SERVER_REWRITE_PHASE	if、return
NGX_HTTP_FIND_CONFIG_PHASE	
NGX_HTTP_REWRITE_PHASE	同server
NGX_HTTP_POST_REWRITE_PHASE	
NGX_HTTP_PREACCESS_PHASE	limit_conn
NGX_HTTP_ACCESS_PHASE	auth_basic_user_file
NGX_HTTP_POST_ACCESS_PHASE	
NGX_HTTP_TRY_FILES_PHASE	try_files
NGX_HTTP_CONTENT_PHASE	index
NGX_HTTP_LOG_PHASE	access_log

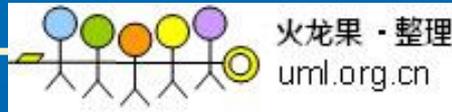
http请求选择location流程



```
server {  
    listen localhost:80; listen 8000;  
    server_name zlddata.com zlddata.cn;  
    location /static { ... } location / { ... }  
}  
Server {  
    listen 80;  
    server_name fuzhong.pub; ...
```

寻找location时仅在NGX_HTTP_FIND_CONFIG_PHASE阶段进行，该阶段前会有post_read和server_rewrite阶段

http模块工作流程



master进程

读取配置文件

打开监听socket

启动worker进程

监控进程状态与信号命令

event模块解析
event {}配置

http模块解析
http {}配置

stream模块解析
stream {}配置

http模块解析配置, 注册 handler

建立 location树

模块回调方法注册 handler

监听端口处理, 如 reuseport

worker进程

接收http包头

调用HTTP模块11个阶段的内容处理方法

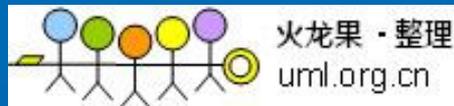
发送HTTP响应

调用HTTP过滤模块的过滤header和过滤body方法

处理事件

处理信号命令

HTTP模块间的配合



接收HTTP头部

preaccess阶段

limit_conn模块

limit_req模块

access阶段

auth_basic模块

access模块

content阶段

http内容模块4

http内容模块5

header过滤模块

image_filter

gzip

发送HTTP头部

body过滤模块

image_filter

gzip

发送HTTP包体

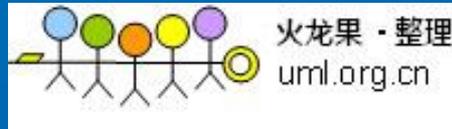
```
limit_req zone=req_one  
burst=120;
```

```
limit_conn c_zone 1;
```

```
satisfy any;  
allow 192.168.1.0/32;  
auth_basic_user_file  
access.pass;
```

```
gzip on;  
image_filter resize 80 80;
```

模块变量



Ngix启动

提供变量的模块

Preconfiguration中定义新的变量

解析出变量的方法

变量名

HTTP头部
读取完毕

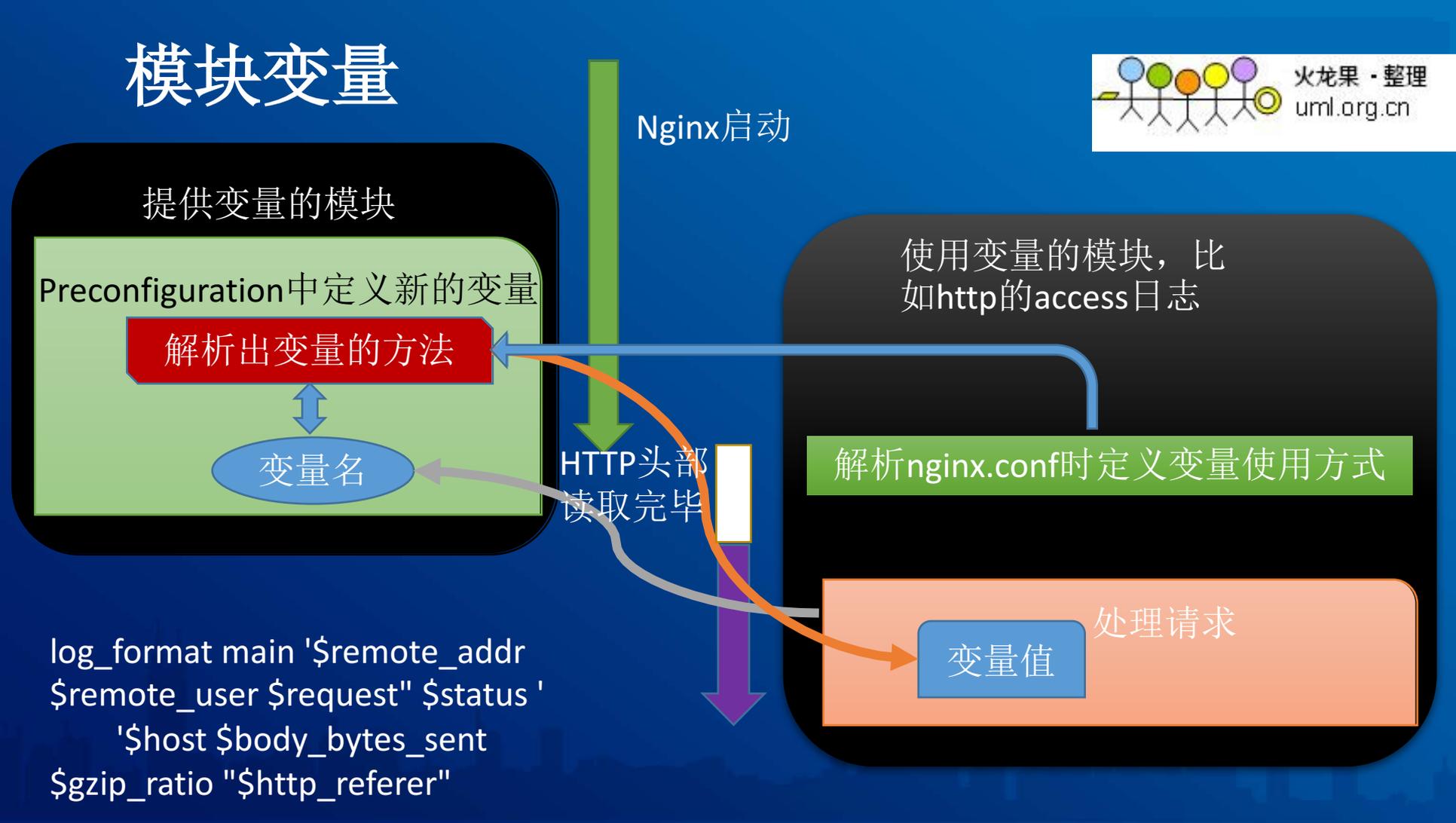
使用变量的模块，比如http的access日志

解析nginx.conf时定义变量使用方式

处理请求

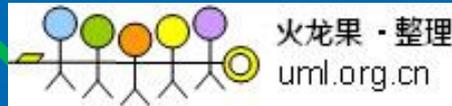
变量值

```
log_format main '$remote_addr  
$remote_user $request' $status '  
$host $body_bytes_sent  
$gzip_ratio "$http_referer"
```

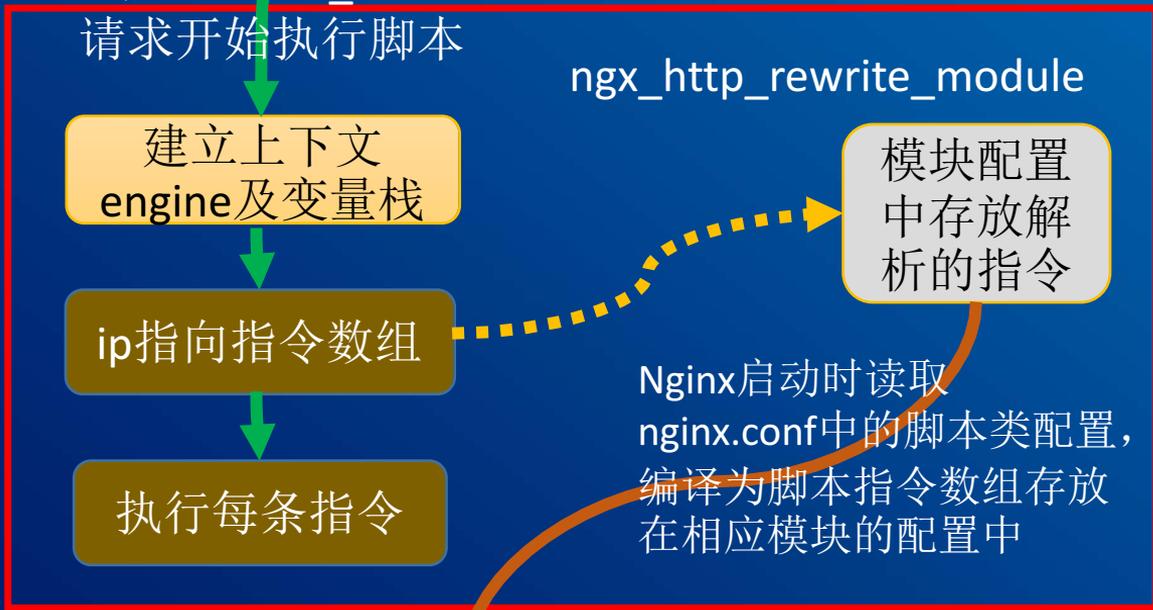


脚本script

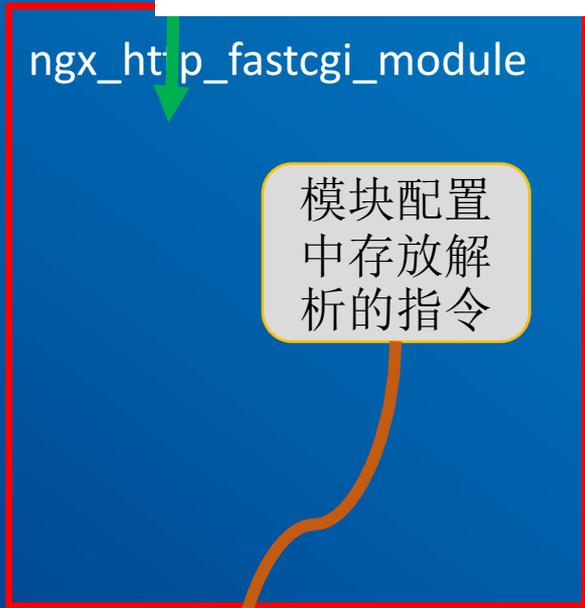
CONTENT_PHASE
阶段执行脚本



2个REWRITE_PHASE
请求开始执行脚本



ngx_http_fastcgi_module



```
try_files $uri $uri/ /index.php?$args;  
return 401 "Access denied because token is expired";  
if ($scheme != "https") {  
    rewrite ^ https://www.mydomain.com$uri permanent;  
}
```

```
uwsgi_store /data/www$original_uri;  
uwsgi_pass localhost:9000;
```

决定模块顺序auto/modules



火龙果·整理
uml.org.cn

```
# the filter order is important
# ngx_http_write_filter
# ngx_http_header_filter
# ngx_http_chunked_filter
# ngx_http_v2_filter
# ngx_http_range_header_filter
# ngx_http_gzip_filter
# ngx_http_postpone_filter
# ngx_http_ssi_filter
# ngx_http_charset_filter

# the module order is important
# ngx_http_static_module
# ngx_http_gzip_static_module
# ngx_http_dav_module
# ngx_http_autoindex_module
# ngx_http_index_module
# ngx_http_random_index_module

# ngx_http_access_module
# ngx_http_realip_module
```



基本模块顺序



过滤模块顺序



3rd模块顺序



objs/nginx_modules.c确认顺序

DEBUG级别日志的阅读



火龙果·整理
uml.org.cn

找到请求

http request line: "GET /
"从uri中寻找请求起始
LOG, http header: "" 确
认头部, http header done
确认所有HTTP头部接收完毕



找到处理它的location

test location: "static "看
看哪些server和location尝试
处理请求, using
configuration " /" 最后又
是哪个location处理了请求

处理请求的模块

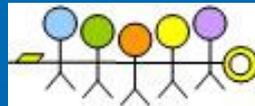
rewrite phase: 2了解经过了
哪些http阶段, http script
copy:" " 找到脚本变量值,
http filename找到静态文件,
uwsgi param找到上游头部



发送HTTP响应

HTTP/1.1 200 OK找到响应
确认头部, http gzip filter
确认过滤模块的处理, write
new buf 确认发送了多少字
节

低投入高产出：理解Nginx



火龙果 · 整理
uml.org.cn

01

Step by step教程

亲自动手

02

按照方向学习某一领域 方面的知识

在stackoverflow上寻找答案

在官方及其他牛人博客上寻找某一场景下的专题技术分析

03

学习Nginx架构

熟悉你需要运行的操作系统

理解产品的场景、常用技术框架，如Http协议、浏览器特性等

理解高并发相关的性能挖掘知识，如零拷贝、DirectIO、TCP状态机等

04

阅读源码

Gdb调试