

深入理解Nginx

模块开发与架构解析

第 2 版

陶辉 著

Understanding Nginx

Modules Development and Architecture Resolving (Second Edition)

- Nginx模块开发领域里程碑之作的升级版，多位权威专家联袂推荐
- 深度还原Nginx设计思想，揭示快速开发简单高效Nginx模块的技巧；透彻解析Nginx架构，拓展开发高性能Web服务器的思路



机械工业出版社
China Machine Press

深入理解Nginx

模块开发与架构解析

第2版

Understanding Nginx

Modules Development and Architecture Resolving (Second Edition)

陶辉 著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

深入理解 Nginx: 模块开发与架构解析 / 陶辉著. —2 版. —北京: 机械工业出版社, 2016.2
(Linux/Unix 技术丛书)

ISBN 978-7-111-52625-4

I. 深… II. 陶… III. 互联网络 - 网络服务器 IV. TP368.5

中国版本图书馆 CIP 数据核字 (2016) 第 010106 号

深入理解 Nginx: 模块开发与架构解析 (第 2 版)

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 余 洁

责任校对: 殷 虹

印 刷: 中国电影出版社印刷厂

版 次: 2016 年 2 月第 2 版第 1 次印刷

开 本: 186mm×240mm 1/16

印 张: 40

书 号: ISBN 978-7-111-52625-4

定 价: 99.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88379426 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzit@hzbook.com

版权所有 • 侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

为什么要写这本书

自第 1 版发行以来，笔者很欣慰得到了广大读者的认可。本书一直致力于说明开发 Nginx 模块的必备知识，然而由于 Nginx 功能繁多且性能强大，以致必须要了解的基本技能也很庞杂，而第 1 版成书匆忙，缺失了几个进阶的技巧描述（例如如何使用变量、slab 共享内存等），因此决定在第 1 版的基础上进一步完善。

事实上，我们总能在 `nginx.conf` 配置文件中看到各种带着 `$` 符号的变量，只要修改带着变量的这一行配置，就可以不用编译、部署而使得 Nginx 具备新功能，这些支持变量的 Nginx 模块提供了极为灵活的功能，第 2 版通过新增的第 15 章详细介绍了如何在模块中支持 HTTP 变量，包括如何在代码中使用其他 Nginx 模块提供的变量，以及如何定义新的变量供 `nginx.conf` 和其他第三方模块使用等。第 16 章介绍了 slab 共享内存，这是一套适用于小块内存快速分配释放的内存管理方式，它非常高效，分配与释放速度都是以纳秒计算的，常用于多个 worker 进程之间的通信，这比第 14 章介绍的原始的共享内存通信方式要先进很多。第 16 章不仅详细介绍了它的实现方式，也探讨了它的优缺点，比如，如果模块间要共享的单个对象常常要消耗数 KB 的空间，这时就需要修改它的实现（例如增大定义的 slab 页大小），以避免内存的浪费等。

Nginx 内存池在第 1 版中只是简单带过，第 2 版中新增了 8.7 节介绍了内存池的实现细节，以帮助读者用好最基础的内存池功能。

此外，很多读者反馈需要结合 TCP 来谈谈 Nginx，因此在 9.10 节中笔者试图在不陷入 Linux 内核细节的情况下，简要介绍了 TCP 以清晰了解 Nginx 的事件框架，了解 Nginx 的高并发能力。

这一版新增的第 15 章的样例代码可以从 <http://nginx.taohui.org.cn> 站点上下载。

因笔者工作繁忙，以致第 2 版拖稿严重，读者的邮件也无法及时回复，非常抱歉。从这

版开始会把曾经的回复整理后放在网站上，想必这比回复邮件要更有效率些。

读者对象

本书适合以下读者阅读。

- 对 Nginx 及如何将它搭建成一个高性能的 Web 服务器感兴趣的读者。
- 希望通过开发特定的 HTTP 模块实现高性能 Web 服务器的读者。
- 希望了解 Nginx 的架构设计，学习其怎样充分使用服务器上的硬件资源的读者。
- 了解如何快速定位、修复 Nginx 中深层次 Bug 的读者。
- 希望利用 Nginx 提供的框架，设计出任何基于 TCP 的、无阻塞的、易于扩展的服务器的读者。

背景知识

如果仅希望了解怎样使用已有的 Nginx 功能搭建服务器，那么阅读本书不需要什么先决条件。但如果希望通过阅读本书的第二、第三两部分，来学习 Nginx 的模块开发和架构设计技巧时，则必须了解 C 语言的基本语法。在阅读本书第三部分时，需要读者对 TCP 有一个基本的了解，同时对 Linux 操作系统也应该有简单的了解。

如何阅读本书

我很希望将本书写成一本“step by step”式（循序渐进式）的书籍，因为这样最能节省读者的时间，然而，由于 3 个主要写作目的想解决的问题都不是那么简单，所以这本书只能做一个折中的处理。

在第一部分的前两章中，将只探讨如何使用 Nginx 这一个问题。阅读这一部分的读者不需要了解 C 语言，就可以学习如何部署 Nginx，学习如何向其中添加各种官方、第三方的功能模块，如何通过修改配置文件来更改 Nginx 及各模块的功能，如何修改 Linux 操作系统上的参数来优化服务器性能，最终向用户提供企业级的 Web 服务器。这一部分介绍配置项的方式，更偏重于领着对 Nginx 还比较陌生的读者熟悉它，通过了解几个基本 Nginx 模块的配置修改方式，进而使读者可以通过查询官网、第三方网站来了解如何使用所有 Nginx 模块的用法。

在第二部分的第 3 章～第 7 章中，都是以例子来介绍 HTTP 模块的开发方式的，这里有些接近于“step by step”的学习方式，我在写作这一部分时，会通过循序渐进的方式使读者能够快速上手，同时会穿插着介绍其常见用法的基本原理。

在第三部分，将开始介绍 Nginx 的完整框架，阅读到这里将会了解第二部分中 HTTP 模块为何以此种方式开发，同时将可以轻易地开发 Nginx 模块。这一部分并不仅仅满足于阐述

Nginx 架构，而是会探讨其为何如此设计，只有这样才能抛开 HTTP 框架、邮件代理框架，实现一种新的业务框架、一种新的模块类型。

对于 Nginx 的使用还不熟悉的读者应当从第 1 章开始学习，前两章将帮助你快速了解 Nginx。

使用过 Nginx，但对如何开发 Nginx 的 HTTP 模块不太了解的读者可以直接从第 3 章开始学习，在这一章阅读完后，即可编写一个功能大致完整的 HTTP 模块。然而，编写企业级的模块必须阅读完第 4 章才能做到，这一章将会介绍编写产品线上服务器程序时必备的 3 个手段。第 5 章举例说明了两种编写复杂 HTTP 模块的方式，在第三部分会对这两个方式有进一步的说明。第 6 章介绍一种特殊的 HTTP 模块——HTTP 过滤模块的编写方法。第 7 章探讨基础容器的用法，这同样是复杂模块的必备工具。

如果读者对于普通 HTTP 模块的编写已经很熟悉，想深入地实现更为复杂的 HTTP 模块，或者想了解邮件代理服务器的设计与实现，或者希望编写一种新的处理其他协议的模块，或者仅仅想了解 Nginx 的架构设计，都可以直接从第 8 章开始学习，这一章会从整体上系统介绍 Nginx 的模块式设计。第 9 章的事件框架是 Nginx 处理 TCP 的基础，这一章无法跳过。阅读第 8 章、第 9 章时可能会遇到许多第 7 章介绍过的容器，这时可以回到第 7 章查询其用法和意义。第 10 章~第 12 章在介绍 HTTP 框架，通过这 3 章的学习会对 HTTP 模块的开发有深入的了解，同时可以学习 HTTP 框架的优秀设计。第 13 章简单介绍了邮件代理服务器的设计，它近似于简化版的 HTTP 框架。第 14 章介绍了进程间同步的工具。第 15 章介绍了 HTTP 变量，包括如何使用已有变量、支持用户在 `nginx.conf` 中修改变量的值、支持其他模块开发者使用自己定义的变量等。第 16 章介绍了 slab 共享内存，该内存极为高效，可用于多个 worker 进程间的通信。

为了不让读者陷入代码的“汪洋大海”中，在本书中大量使用了图表，这样可以使读者快速、大体地了解流程和原理，在这基础上，如果读者还希望了解代码是如何实现的，可以针对性地阅读源代码中的相应方法。在代码的关键地方会通过添加注释的方式加以说明。希望这种方式能够帮助读者减少阅读花费的时间，更快、更好地把握住 Nginx，同时深入到细节中。

写作本书第 1 版时，Nginx 的最新稳定版本是 1.0.14，所以当时是基于此版本来写作的。截止到第 2 版完成时，Nginx 的稳定版本已经上升到了 1.8.0。但这不会对本书的阅读造成困惑，笔者验证过示例代码，均可以运行在最新版本的 Nginx 中，这是因为本书主要是在介绍 Nginx 的基本框架代码，以及怎样使用这些框架代码开发新的 Nginx 模块。在这些基本框架代码中，Nginx 一般不会做任何改变，否则已有的大量 Nginx 模块将无法工作，这种损失是不可承受的。而且 Nginx 框架为具体的功能模块提供了足够的灵活性，修改功能时很少需要修改

框架代码。

Nginx 是跨平台的服务器，然而这本书将只针对于最常见的 Linux 操作系统进行分析，这样做一方面是篇幅所限，另一方面则是本书的写作目的主要在于告诉大家如何基于 Nginx 编写代码，而不是怎样在一个具体的操作系统上修改配置使用 Nginx。因此，即使本书以 Linux 系统为代表讲述 Nginx，也不会影响使用其他操作系统的读者阅读，操作系统的差别相对于本书内容的影响实在是非常小。

勘误和支持

由于作者的水平有限，加之编写的时间也很仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。为此，我特意创建了一个在线支持与应急方案的二级站点：<http://nginx.weebly.com>。读者可以将书中的错误发布在 Bug 勘误表页面中，同时如果读者遇到任何问题，也可以访问 Q&A 页面，我将尽量在线上为读者提供最满意的解答。书中的全部源文件都将发布在这个网站上，我也会将相应的功能更新及时发布出来。如果你有更多的宝贵意见，也欢迎你发送邮件至我的邮箱 russelltao@foxmail.com，期待能够听到读者的真挚反馈。

致谢

我首先要感谢 Igor Sysoev，他在 Nginx 设计上展现的功力令人折服，正是他的工作成果才有了本书诞生的意义。

lisa 是机械工业出版社华章公司的优秀编辑，非常值得信任。在这半年的写作过程中，她花费了很多时间、精力来阅读我的书稿，指出了许多文字上和格式上的错误，她提出的建议都大大提高了本书的可读性。

在这半年时间里，一边工作一边写作给我带来了很大的压力，所以我要感谢我的父母在生活上对我无微不至的照顾，使我可以全力投入到写作中。繁忙的工作之余，写作又占用了休息时间的绝大部分，感谢我的太太毛业勤对我的体谅和鼓励，让我始终以高昂的斗志投入到本书的写作中。

感谢我工作中的同事们，正是在与他们一起战斗在一线的日子里，我才不断地对技术有新地感悟；正是那些充满激情的岁月，才使得我越来越热爱服务器技术的开发。

谨以此书，献给我最亲爱的家人，以及众多热爱 Nginx 的朋友。

陶辉

2015 年 10 月

Contents 目 录

前 言

第一部分 Nginx 能帮我们做什么

第 1 章 研究 Nginx 前的准备工作 2

- 1.1 Nginx 是什么 2
- 1.2 为什么选择 Nginx 5
- 1.3 准备工作 7
 - 1.3.1 Linux 操作系统 7
 - 1.3.2 使用 Nginx 的必备软件 7
 - 1.3.3 磁盘目录 8
 - 1.3.4 Linux 内核参数的优化 9
 - 1.3.5 获取 Nginx 源码 10
- 1.4 编译安装 Nginx 11
- 1.5 configure 详解 11
 - 1.5.1 configure 的命令参数 11
 - 1.5.2 configure 执行流程 18
 - 1.5.3 configure 生成的文件 21
- 1.6 Nginx 的命令行控制 23
- 1.7 小结 27

第 2 章 Nginx 的配置 28

- 2.1 运行中的 Nginx 进程间的关系 28

2.2 Nginx 配置的通用语法 31

- 2.2.1 块配置项 31
- 2.2.2 配置项的语法格式 32
- 2.2.3 配置项的注释 33
- 2.2.4 配置项的单位 33
- 2.2.5 在配置中使用变量 33

2.3 Nginx 服务的基本配置 34

- 2.3.1 用于调试进程和定位问题的配置项 34
- 2.3.2 正常运行的配置项 36
- 2.3.3 优化性能的配置项 37
- 2.3.4 事件类配置项 39

2.4 用 HTTP 核心模块配置一个静态

- Web 服务器 40
- 2.4.1 虚拟主机与请求的分发 41
- 2.4.2 文件路径的定义 45
- 2.4.3 内存及磁盘资源的分配 47
- 2.4.4 网络连接的设置 49
- 2.4.5 MIME 类型的设置 52
- 2.4.6 对客户端请求的限制 53
- 2.4.7 文件操作的优化 54
- 2.4.8 对客户端请求的特殊处理 56

2.4.9 ngx_http_core_module 模块 提供的变量	57
2.5 用 HTTP proxy module 配置一个 反向代理服务器	59
2.5.1 负载均衡的基本配置	61
2.5.2 反向代理的基本配置	63
2.6 小结	66

第二部分 如何编写 HTTP 模块

第 3 章 开发一个简单的 HTTP 模块 .. 68

3.1 如何调用 HTTP 模块	68
3.2 准备工作	70
3.2.1 整型的封装	71
3.2.2 ngx_str_t 数据结构	71
3.2.3 ngx_list_t 数据结构	71
3.2.4 ngx_table_elt_t 数据结构	75
3.2.5 ngx_buf_t 数据结构	75
3.2.6 ngx_chain_t 数据结构	77
3.3 如何将自己的 HTTP 模块 编译进 Nginx	77
3.3.1 config 文件的写法	77
3.3.2 利用 configure 脚本将定制的 模块加入到 Nginx 中	78
3.3.3 直接修改 Makefile 文件	81
3.4 HTTP 模块的数据结构	82
3.5 定义自己的 HTTP 模块	86
3.6 处理用户请求	89
3.6.1 处理方法的返回值	89
3.6.2 获取 URI 和参数	92
3.6.3 获取 HTTP 头部	94

3.6.4 获取 HTTP 包体	97
3.7 发送响应	99
3.7.1 发送 HTTP 头部	99
3.7.2 将内存中的字符串作为包体 发送	101
3.7.3 经典的“Hello World”示例	102
3.8 将磁盘文件作为包体发送	103
3.8.1 如何发送磁盘中的文件	104
3.8.2 清理文件句柄	106
3.8.3 支持用户多线程下载和断点 续传	107
3.9 用 C++ 语言编写 HTTP 模块	108
3.9.1 编译方式的修改	108
3.9.2 程序中的符号转换	109
3.10 小结	110

第 4 章 配置、error 日志和请求

上下文

4.1 http 配置项的使用场景	111
4.2 怎样使用 http 配置	113
4.2.1 分配用于保存配置参数的 数据结构	113
4.2.2 设定配置项的解析方式	115
4.2.3 使用 14 种预设方法解析 配置项	121
4.2.4 自定义配置项处理方法	131
4.2.5 合并配置项	133
4.3 HTTP 配置模型	135
4.3.1 解析 HTTP 配置的流程	136
4.3.2 HTTP 配置模型的内存布局	139
4.3.3 如何合并配置项	142

4.3.4 预设配置项处理方法的工作原理	144	5.3.4 在 process_header 方法中解析包头	171
4.4 error 日志的用法	145	5.3.5 在 finalize_request 方法中释放资源	175
4.5 请求的上下文	149	5.3.6 在 ngx_http_mytest_handler 方法中启动 upstream	175
4.5.1 上下文与全异步 Web 服务器的关系	149	5.4 subrequest 的使用方式	177
4.5.2 如何使用 HTTP 上下文	151	5.4.1 配置子请求的处理方式	177
4.5.3 HTTP 框架如何维护上下文结构	152	5.4.2 实现子请求处理完毕时的回调方法	178
4.6 小结	153	5.4.3 处理父请求被重新激活后的回调方法	179
第 5 章 访问第三方服务	154	5.4.4 启动 subrequest 子请求	179
5.1 upstream 的使用方式	155	5.5 subrequest 执行过程中的主要场景	180
5.1.1 ngx_http_upstream_t 结构体	158	5.5.1 如何启动 subrequest	180
5.1.2 设置 upstream 的限制性参数	159	5.5.2 如何转发多个子请求的响应包体	182
5.1.3 设置需要访问的第三方服务器地址	160	5.5.3 子请求如何激活父请求	185
5.1.4 设置回调方法	161	5.6 subrequest 使用的例子	187
5.1.5 如何启动 upstream 机制	161	5.6.1 配置文件中子请求的设置	187
5.2 回调方法的执行场景	162	5.6.2 请求上下文	188
5.2.1 create_request 回调方法	162	5.6.3 子请求结束时的处理方法	188
5.2.2 reinit_request 回调方法	164	5.6.4 父请求的回调方法	189
5.2.3 finalize_request 回调方法	165	5.6.5 启动 subrequest	190
5.2.4 process_header 回调方法	165	5.7 小结	191
5.2.5 rewrite_redirect 回调方法	167		
5.2.6 input_filter_init 与 input_filter 回调方法	167	第 6 章 开发一个简单的 HTTP 过滤模块	192
5.3 使用 upstream 的示例	168	6.1 过滤模块的意义	192
5.3.1 upstream 的各种配置参数	168	6.2 过滤模块的调用顺序	193
5.3.2 请求上下文	170	6.2.1 过滤链表是如何构成的	194
5.3.3 在 create_request 方法中构造请求	170		

6.2.2	过滤链表的顺序	196	7.5.2	红黑树的特性	220
6.2.3	官方默认 HTTP 过滤模块的 功能简介	197	7.5.3	红黑树的使用方法	222
6.3	HTTP 过滤模块的开发步骤	198	7.5.4	使用红黑树的简单例子	225
6.4	HTTP 过滤模块的简单例子	200	7.5.5	如何自定义添加成员方法	226
6.4.1	如何编写 config 文件	201	7.6	ngx_radix_tree_t 基数树	228
6.4.2	配置项和上下文	201	7.6.1	ngx_radix_tree_t 基数树的 原理	228
6.4.3	定义 HTTP 过滤模块	203	7.6.2	基数树的使用方法	230
6.4.4	初始化 HTTP 过滤模块	204	7.6.3	使用基数树的例子	231
6.4.5	处理请求中的 HTTP 头部	204	7.7	支持通配符的散列表	232
6.4.6	处理请求中的 HTTP 包体	206	7.7.1	ngx_hash_t 基本散列表	232
6.5	小结	206	7.7.2	支持通配符的散列表	235
第 7 章 Nginx 提供的高级数据结构			7.7.3	带通配符散列表的使用例子	241
	结构	207	7.8	小结	245
7.1	Nginx 提供的高级数据结构概述	207	第三部分 深入 Nginx		
7.2	ngx_queue_t 双向链表	209	第 8 章 Nginx 基础架构		
7.2.1	为什么设计 ngx_queue_t 双向链表	209	8.1	Web 服务器设计中的关键约束	249
7.2.2	双向链表的使用方法	209	8.2	Nginx 的架构设计	251
7.2.3	使用双向链表排序的例子	212	8.2.1	优秀的模块化设计	251
7.2.4	双向链表是如何实现的	213	8.2.2	事件驱动架构	254
7.3	ngx_array_t 动态数组	215	8.2.3	请求的多阶段异步处理	256
7.3.1	为什么设计 ngx_array_t 动态数组	215	8.2.4	管理进程、多工作进程设计	259
7.3.2	动态数组的使用方法	215	8.2.5	平台无关的代码实现	259
7.3.3	使用动态数组的例子	217	8.2.6	内存池的设计	259
7.3.4	动态数组的扩容方式	218	8.2.7	使用统一管道过滤器模式的 HTTP 过滤模块	260
7.4	ngx_list_t 单向链表	219	8.2.8	其他一些用户模块	260
7.5	ngx_rbtree_t 红黑树	219	8.3	Nginx 框架中的核心结构体 ngx_cycle_t	260
7.5.1	为什么设计 ngx_rbtree_t 红黑树	219			

8.3.1 ngx_listening_t 结构体	261	9.8 事件驱动框架的处理流程	324
8.3.2 ngx_cycle_t 结构体	262	9.8.1 如何建立新连接	325
8.3.3 ngx_cycle_t 支持的方法	264	9.8.2 如何解决“惊群”问题	327
8.4 Nginx 启动时框架的处理流程	266	9.8.3 如何实现负载均衡	329
8.5 worker 进程是如何工作的	269	9.8.4 post 事件队列	330
8.6 master 进程是如何工作的	271	9.8.5 ngx_process_events_and_timers 流程	331
8.7 ngx_pool_t 内存池	276	9.9 文件的异步 I/O	334
8.8 小结	284	9.9.1 Linux 内核提供的文件异步 I/O	335
第 9 章 事件模块	285	9.9.2 ngx_epoll_module 模块中 实现的针对文件的异步 I/O	337
9.1 事件处理框架概述	286	9.10 TCP 协议与 Nginx	342
9.2 Nginx 事件的定义	288	9.11 小结	347
9.3 Nginx 连接的定义	291	第 10 章 HTTP 框架的初始化	348
9.3.1 被动连接	292	10.1 HTTP 框架概述	349
9.3.2 主动连接	295	10.2 管理 HTTP 模块的配置项	352
9.3.3 ngx_connection_t 连接池	296	10.2.1 管理 main 级别下的配置项	353
9.4 ngx_events_module 核心模块	297	10.2.2 管理 server 级别下的配置项	355
9.4.1 如何管理所有事件模块的 配置项	299	10.2.3 管理 location 级别下的 配置项	358
9.4.2 管理事件模块	300	10.2.4 不同级别配置项的合并	364
9.5 ngx_event_core_module 事件 模块	302	10.3 监听端口的管理	367
9.6 epoll 事件驱动模块	308	10.4 server 的快速检索	370
9.6.1 epoll 的原理和用法	308	10.5 location 的快速检索	370
9.6.2 如何使用 epoll	310	10.6 HTTP 请求的 11 个处理阶段	372
9.6.3 ngx_epoll_module 模块的 实现	312	10.6.1 HTTP 处理阶段的普适规则	374
9.7 定时器事件	320	10.6.2 NGX_HTTP_POST_READ_ PHASE 阶段	375
9.7.1 缓存时间的管理	320	10.6.3 NGX_HTTP_SERVER_ REWRITE_PHASE 阶段	378
9.7.2 缓存时间的精度	323		
9.7.3 定时器的实现	323		

10.6.4	NGX_HTTP_FIND_ CONFIG_PHASE 阶段	378	11.6.3	ngx_http_core_access_phase	409
10.6.5	NGX_HTTP_REWRITE_ PHASE 阶段	378	11.6.4	ngx_http_core_content_ phase	412
10.6.6	NGX_HTTP_POST_ REWRITE_PHASE 阶段	379	11.7	subrequest 与 post 请求	415
10.6.7	NGX_HTTP_PREACCESS_ PHASE 阶段	379	11.8	处理 HTTP 包体	417
10.6.8	NGX_HTTP_ACCESS_ PHASE 阶段	379	11.8.1	接收包体	419
10.6.9	NGX_HTTP_POST_ ACCESS_PHASE 阶段	380	11.8.2	放弃接收包体	425
10.6.10	NGX_HTTP_TRY_FILES_ PHASE 阶段	380	11.9	发送 HTTP 响应	429
10.6.11	NGX_HTTP_CONTENT_ PHASE 阶段	380	11.9.1	ngx_http_send_header	430
10.6.12	NGX_HTTP_LOG_PHASE 阶段	382	11.9.2	ngx_http_output_filter	432
10.7	HTTP 框架的初始化流程	382	11.9.3	ngx_http_writer	435
10.8	小结	384	11.10	结束 HTTP 请求	437
第 11 章 HTTP 框架的执行流程			11.10.1	ngx_http_close_connection	438
11.1	HTTP 框架执行流程概述	386	11.10.2	ngx_http_free_request	439
11.2	新连接建立时的行为	387	11.10.3	ngx_http_close_request	440
11.3	第一次可读事件的处理	388	11.10.4	ngx_http_finalize_ connection	441
11.4	接收 HTTP 请求行	394	11.10.5	ngx_http_terminate_ request	443
11.5	接收 HTTP 头部	398	11.10.6	ngx_http_finalize_request	443
11.6	处理 HTTP 请求	400	11.11	小结	446
11.6.1	ngx_http_core_generic_ phase	406	第 12 章 upstream 机制的设计与实现		
11.6.2	ngx_http_core_rewrite_ phase	408	12.1	upstream 机制概述	448
			12.1.1	设计目的	448
			12.1.2	ngx_http_upstream_t 数据 结构的意义	450
			12.1.3	ngx_http_upstream_conf_t 配置结构体	453
			12.2	启动 upstream	455
			12.3	与上游服务器建立连接	457

12.4	发送请求到上游服务器	460	13.2.3	邮件框架的初始化	506
12.5	接收上游服务器的响应头部	463	13.3	初始化请求	506
12.5.1	应用层协议的两段划分 方式	463	13.3.1	描述邮件请求的 ngx_mail_ session_t 结构体	506
12.5.2	处理包体的 3 种方式	464	13.3.2	初始化邮件请求的流程	509
12.5.3	接收响应头部的流程	465	13.4	接收并解析客户端请求	509
12.6	不转发响应时的处理流程	469	13.5	邮件认证	510
12.6.1	input_filter 方法的设计	469	13.5.1	ngx_mail_auth_http_ctx_t 结构体	510
12.6.2	默认的 input_filter 方法	470	13.5.2	与认证服务器建立连接	511
12.6.3	接收包体的流程	472	13.5.3	发送请求到认证服务器	513
12.7	以下游网速优先来转发响应	473	13.5.4	接收并解析响应	514
12.7.1	转发响应的包头	474	13.6	与上游邮件服务器间的认证 交互	514
12.7.2	转发响应的包体	477	13.6.1	ngx_mail_proxy_ctx_t 结构体	516
12.8	以上游网速优先来转发响应	481	13.6.2	向上游邮件服务器发起 连接	516
12.8.1	ngx_event_pipe_t 结构体的 意义	481	13.6.3	与邮件服务器认证交互的 过程	518
12.8.2	转发响应的包头	485	13.7	透传上游邮件服务器与客户 端间的流	520
12.8.3	转发响应的包体	487	13.8	小结	524
12.8.4	ngx_event_pipe_read_ upstream 方法	489			
12.8.5	ngx_event_pipe_write_to_ downstream 方法	494			
12.9	结束 upstream 请求	496			
12.10	小结	499			
第 13 章	邮件代理模块	500	第 14 章	进程间的通信机制	525
13.1	邮件代理服务器的功能	500	14.1	概述	525
13.2	邮件模块的处理框架	503	14.2	共享内存	526
13.2.1	一个请求的 8 个独立处理 阶段	503	14.3	原子操作	530
13.2.2	邮件类模块的定义	504	14.3.1	不支持原子库下的原子 操作	530
			14.3.2	x86 架构下的原子操作	531

14.3.3 自旋锁	533	15.2.5 如何解析变量	573
14.4 Nginx 频道	535	15.3 定义内部变量	576
14.5 信号	538	15.4 外部变量与脚本引擎	577
14.6 信号量	540	15.4.1 相关数据结构	578
14.7 文件锁	541	15.4.2 编译“set”脚本	581
14.8 互斥锁	544	15.4.3 脚本执行流程	586
14.8.1 文件锁实现的 ngx_ shmtx_t 锁	546	15.5 小结	589
14.8.2 原子变量实现的 ngx_ shmtx_t 锁	548		
14.9 小结	553	第 16 章 slab 共享内存	590
第 15 章 变量	554	16.1 操作 slab 共享内存的方法	590
15.1 使用内部变量开发模块	555	16.2 使用 slab 共享内存池的例子	592
15.1.1 定义模块	556	16.2.1 共享内存中的数据结构	593
15.1.2 定义 http 模块加载方式	557	16.2.2 操作共享内存中的红黑树 与链表	595
15.1.3 解析配置中的变量	558	16.2.3 解析配置文件	600
15.1.4 处理请求	560	16.2.4 定义模块	603
15.2 内部变量工作原理	561	16.3 slab 内存管理的实现原理	605
15.2.1 何时定义变量	561	16.3.1 内存结构布局	607
15.2.2 相关数据结构详述	564	16.3.2 分配内存流程	613
15.2.3 定义变量的方法	572	16.3.3 释放内存流程	617
15.2.4 使用变量的方法	572	16.3.4 如何使用位操作	619
		16.3.5 slab 内存池间的管理	624
		16.4 小结	624

第一部分 *Part 1*

Nginx 能帮我们做什么

- 第 1 章 研究 Nginx 前的准备工作
 - 第 2 章 Nginx 的配置
- 

研究 Nginx 前的准备工作

2012 年, Nginx 荣获年度云计算开发奖 (2012 Cloud Award for Developer of the Year), 并成长为世界第二大 Web 服务器。全世界流量最高的前 1000 名网站中, 超过 25% 都使用 Nginx 来处理海量的互联网请求。Nginx 已经成为业界高性能 Web 服务器的代名词。

那么, 什么是 Nginx? 它有哪些特点? 我们选择 Nginx 的理由是什么? 如何编译安装 Nginx? 这种安装方式背后隐藏的又是什么样的思想呢? 本章将会回答上述问题。

1.1 Nginx 是什么

人们在了解新事物时, 往往习惯通过类比来帮助自己理解事物的概貌。那么, 我们在学习 Nginx 时也采用同样的方式, 先来看看 Nginx 的竞争对手——Apache、Lighttpd、Tomcat、Jetty、IIS, 它们都是 Web 服务器, 或者叫做 WWW (World Wide Web) 服务器, 相应地也都具备 Web 服务器的基本功能: 基于 REST 架构风格^①, 以统一资源描述符 (Uniform Resource Identifier, URI) 或者统一资源定位符 (Uniform Resource Locator, URL) 作为沟通依据, 通过 HTTP 为浏览器等客户端程序提供各种网络服务。然而, 由于这些 Web 服务器在设计阶段就受到许多局限, 例如当时的互联网用户规模、网络带宽、产品特点等局限, 并且各自的定位与发展方向都不尽相同, 使得每一款 Web 服务器的特点与应用场合都很鲜明。

Tomcat 和 Jetty 面向 Java 语言, 先天就是重量级的 Web 服务器, 它的性能与 Nginx 没有可比性, 这里略过。

① 参见 Roy Fielding 博士的论文《Architectural Styles and the Design of Network-based Software Architectures》, 可在 <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> 查看原文。