

<<深入理解Nginx>>

图书基本信息

书名：<<深入理解Nginx>>

13位ISBN编号：9787111414780

10位ISBN编号：7111414780

出版时间：2013-4-15

出版时间：陶辉 机械工业出版社 (2013-03出版)

作者：陶辉

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<深入理解Nginx>>

前言

为什么要写这本书当我试图在产品的关键位置设计一个高性能Web服务器时，我选择使用成熟的Nginx。

选择它的理由为：首先，它对服务器性能上的挖掘已经达到了很高水平，它能尽量使不同的硬件（包括网卡、硬盘、不同的CPU核心）并发运行，同时软件中又没有阻塞进程使之睡眠的代码，从性能上来说，它可以挑战任何服务器。

其次，完全基于事件驱动的服务器开发效率往往很不理想，它们要处理的事件过于底层化、细节化，这使得各功能模块无法聚焦于业务，最终产品的功能都较为单一，不会有丰富的可选功能。

但Nginx却不然，由于它在软件架构上具有优秀的设计，使得Nginx完全由许多简单的模块构成，各模块（特别是HTTP模块）不用介入底层细节，在尽享分阶段、无阻塞的事件驱动架构下，可以专注于业务功能的实现，这样最终为Nginx带来了大量的官方、第三方的功能模块，使得功能同样强大的Nginx在产品核心位置上足以担当重任，经受住海量请求的考验。

当Nginx已有模块提供的功能不能完全实现我的所有业务需求时，我可以在Nginx的后端再搭建一个实现了缺失功能的非Nginx服务器，将Nginx无法实现的请求反向代理到这台服务器上处理。

但这样也有一定的弊端，首先增大了处理请求的开销，其次后端服务器的设计仍然制约着总体性能（它依然需要解决Nginx解决过的无阻塞问题，那样才能像Nginx一样高效），这样做仅适用于对性能要求不高的场景。

唯有开发一个实现了所需功能的自定义Nginx模块嵌入到Nginx代码中，才能让自己的业务像Nginx一样充分挖掘服务器的硬件资源，及时地响应百万级别的并发TCP连接。

当我在开发Nginx模块之前，试图在市场上找到一本关于Nginx模块开发的书籍（无论是中文还是英文）时却一无所获。

我只能找到如何使用Nginx及其已有模块的书籍。

为了开发Nginx模块，我只能通过阅读Nginx极度缺少注释的源代码，并分析各种官方Nginx模块来逐步还原其设计思想，反复尝试、验证着怎样的模块能够使用Nginx的基础架构，和丰富的跨平台工具方法，同时符合Nginx设计思想，使Nginx拥有媲美Linux内核的一流效率。

这个过程耗费了我很多的精力，因此，我希望今后的Nginx使用者、开发者在遇到同样的问题时，不至于还要很痛苦地阅读源代码来找到模块开发方法，而是简单地按照章节查阅本书，就可以快速找到怎样简单、高效地开发Nginx模块，把精力放在业务的实现上。

这是我写这本书的第一个目的。

当我们产品中运行的Nginx出现了问题时，往往是通过找到错误的配置项、使用方式来解决的，这样也的确能够修复大部分问题。

但是更深层次的问题，或者是使用场景比较偏僻，抑或是Nginx自身代码考虑得不够全面时，这些问题往往只能由那些花费大量精力研究Nginx源代码的工程师来解决。

我写作本书的第二个目的是希望通过透彻地解析Nginx架构，帮助读者深入理解Nginx，既能够正确地使用它，也能在它出现任何问题时找到根本原因，进而用最合适的方法修复或者回避问题。

Nginx是一个优秀的事件驱动框架，虽然它在HTTP的处理上非常出色，但它绝不仅仅用于Web服务器。

Nginx非常适合开发在传输层以TCP对外提供服务的服务器程序。

基于Nginx框架开发程序有5个优势：1) Nginx将网络、磁盘及定时器等异步事件的驱动都做了非常好的封装，基于它开发将可以忽略这些事件处理的细节。

2) Nginx封装了许多平台无关的接口、容器，适用于跨平台开发。

3) 优秀的模块化设计，使得开发者可以轻易地复用各种已有的模块，其中既包括基本的读取配置、记录日志等模块，也包括处理请求的诸如HTTP、mail等高级功能模块。

4) Nginx是作为服务器来设计其框架的，因此，它在服务器进程的管理上相当出色，基于它开发服务器程序可以轻松地实现程序的动态升级，子进程的监控、管理，配置项的动态修改生效等。

5) Nginx充分考虑到各操作系统所擅长的“绝活”，能够使用特殊的系统调用更高效地完成任

<<深入理解Nginx>>

绝不会去使用低效的通用接口。

尤其对于Linux操作系统，Nginx不遗余力地做了大量优化。

当我们期望编写一款能够以低负载处理高并发请求并且主要处理基于TCP的服务器程序时，推荐选择Nginx，它可能会带给我们意外的惊喜。

这本书的第三部分，将通过分析Nginx的内部架构，帮助读者了解怎样基于Nginx开发高效的TCP服务器程序：通过开发一种新的模块类型，实现一种新的功能框架来提供极佳的扩展性，使得功能子模块仅关注于业务的开发，忽视底层事件的处理。

这是我写作本书的第三个目的。

除了这3个主要目的外，我还希望通过这本书向大家展示Nginx在服务器开发上的许多巧妙设计，它们或在抽象设计上精妙，或通过操作系统精确、节省地使用硬件资源，这些细节之处的设计都体现了Igor Sysoev的不凡功底。

即使我们完全不使用Nginx，学习这些技巧也将有助于我们服务器编程水平的提升。

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、第三部分，来学习Nginx的模块开发和架构设计技巧，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，更偏重于带领对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，阅读到这里时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出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种方式有进一步的说明。

<<深入理解Nginx>>

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、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章简单地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。

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。

关键地方会直接给出代码，并添加注释加以说明。

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Nginx，同时深入到细节中。

在本书开始写作时，由于Nginx的最新稳定版本是1.0.14，所以本书是基于此版本来编写的。

截止到本书编写完成时，Nginx的稳定版本已经上升到了1.2.4。

但这不会对本书的阅读造成困扰，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，而不是介绍Nginx的某些功能。

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，这种损失也是不可承受的。

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读者如何基于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来使用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，因为操作系统的差别对阅读本书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。

读者可以将书中的错误发布在Bug勘误表页面中，同时如果你遇到任何问题，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，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让本书的诞生有了意义。

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，指出了许多文字和格式上的错误，她提出的建议大大提高了本书的可读性。

在这半年时间内，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，正是在与他们一起工作的日子里，我才不断地对技术有新的感悟；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

<<深入理解Nginx>>

内容概要

本书是阿里巴巴资深Nginx技术专家呕心沥血之作，是作者多年的经验结晶，也是目前市场上唯一一本通过还原Nginx设计思想，剖析Nginx架构来帮助读者快速高效开发HTTP模块的图书。

本书首先通过介绍官方Nginx的基本用法和配置规则，帮助读者了解一般Nginx模块的用法，然后重点介绍如何开发HTTP模块（含HTTP过滤模块）来得到定制的Nginx，其中包括开发一个功能复杂的模块所需要了解的各种知识，如Nginx的基础数据结构、配置项的解析、记录日志的工具以及upstream、subrequest的使用方法等。

在此基础上，综合Nginx框架代码分析Nginx的架构，介绍其设计理念和技巧，进一步帮助读者自由、有效地开发出功能丰富、性能一流的Nginx模块。

<<深入理解Nginx>>

作者简介

陶辉，思科后台工程师，从事服务端开发近十年，擅长Linux分布式架构下的海量数据处理，擅长C/C++开发的高性能高吞吐量网络服务，曾任思科DMS后台架构设计工作，曾在腾讯QQ空间后台个人信息中心、个人档、漂流瓶、空间日志、花藤、好友买卖等项目中担任服务器设计与开发工作，曾在华为中央软件部综合网管平台担任北向接口设计开发工作。拥有丰富的Linux高性能服务器开发经验，丰富的云存储系统开发经验，目前关注云文档管理系统和Nginx的再开发。

<<深入理解Nginx>>

书籍目录

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 / 4 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 / 11 1.4 编译安装Nginx / 11 1.5 configure详解 / 11 1.5.1 configure的命令参数 / 12 1.5.2 configure执行流程 / 18 1.5.3 configure生成的文件 / 22 1.6 Nginx的命令行控制 / 24 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 / 38 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 / 41 2.4.1 虚拟主机与请求的分发 / 42 2.4.2 文件路径的定义 / 45 2.4.3 内存及磁盘资源的分配 / 48 2.4.4 网络连接的设置 / 50 2.4.5 MIME类型的设置 / 53 2.4.6 对客户端请求的限制 / 54 2.4.7 文件操作的优化 / 55 2.4.8 对客户端请求的特殊处理 / 57 2.4.9 ngx_http_core_module模块提供的变量 / 59 2.5 用HTTPproxymodule配置一个反向代理服务器 / 60 2.5.1 负载均衡的基本配置 / 62 2.5.2 反向代理的基本配置 / 64 2.6 小结 / 68 第二部分 如何编写HTTP模块 第3章 开发一个简单的HTTP模块 / 70 3.1 如何调用HTTP模块 / 70 3.2 准备工作 / 72 3.2.1 整型的封装 / 72 3.2.2 ngx_str_t数据结构 / 73 3.2.3 ngx_list_t数据结构 / 73 3.2.4 ngx_table_elt_t数据结构 / 77 3.2.5 ngx_buf_t数据结构 / 77 3.2.6 ngx_chain_t数据结构 / 79 3.3 如何将自己的HTTP模块编译进Nginx / 79 3.3.1 config文件的写法 / 80 3.3.2 利用configure脚本将定制模块加入到Nginx中 / 80 3.3.3 直接修改Makefile文件 / 84 3.4 HTTP模块的数据结构 / 85 3.5 定义自己的HTTP模块 / 88 3.6 处理用户请求 / 92 3.6.1 处理方法的返回值 / 92 3.6.2 获取URI和参数 / 95 3.6.3 获取HTTP头部 / 98 3.6.4 获取HTTP包体 / 101 3.7 发送响应 / 102 3.7.1 发送HTTP头部 / 102 3.7.2 将内存中的字符串作为包体发送 / 104 3.7.3 经典的“HelloWorld”示例 / 106 3.8 将磁盘文件作为包体发送 / 107 3.8.1 如何发送磁盘中的文件 / 107 3.8.2 清理文件句柄 / 110 3.8.3 支持用户多线程下载和断点续传 / 111 3.9 用C++语言编写HTTP模块 / 112 3.9.1 编译方式的修改 / 112 3.9.2 程序中的符号转换 / 114 3.10 小结 / 114 第4章 配置、error日志和请求上下文 / 115 4.1 http配置项的使用场景 / 115 4.2 怎样使用http配置 / 117 4.2.1 分配用于保存配置参数的数据结构 / 117 4.2.2 设定配置项的解析方式 / 119 4.2.3 使用14种预设方法解析配置项 / 125 4.2.4 自定义配置项处理方法 / 136 4.2.5 合并配置项 / 137 4.3 HTTP配置模型 / 140 4.3.1 解析HTTP配置的流程 / 141 4.3.2 HTTP配置模型的内存布局 / 144 4.3.3 如何合并配置项 / 147 4.3.4 预设配置项处理方法的工作原理 / 149 4.4 error日志的用法 / 150 4.5 请求的上下文 / 155 4.5.1 上下文与全异步Web服务器的关系 / 155 4.5.2 如何使用HTTP上下文 / 156 4.5.3 HTTP框架如何维护上下文结构 / 157 4.6 小结 / 158 第5章 访问第三方服务 / 159 5.1 upstream的使用方式 / 160 5.1.1 ngx_http_upstream_t结构体 / 163 5.1.2 设置upstream的限制性参数 / 164 5.1.3 设置需要访问的第三方服务器地址 / 165 5.1.4 设置回调方法 / 166 5.1.5 如何启动upstream机制 / 166 5.2 回调方法的执行场景 / 167 5.2.1 create_request回调方法 / 167 5.2.2 reinit_request回调方法 / 169 5.2.3 finalize_request回调方法 / 170 5.2.4 process_header回调方法 / 171 5.2.5 rewrite_redirect回调方法 / 172 5.2.6 input_filter_init与input_filter回调方法 / 172 5.3 使用upstream的示例 / 173 5.3.1 upstream的各种配置参数 / 174 5.3.2 请求上下文 / 175 5.3.3 在create_request方法中构造请求 / 176 5.3.4 在process_header方法中解析包头 / 177 5.3.5 在finalize_request方法中释放资源 / 180 5.3.6 在ngx_http_mytest_handler方法中启动upstream / 181 5.4 subrequest的使用方式 / 183 5.4.1 配置子请求的处理方式 / 183 5.4.2 实现子请求处理完毕时的回调方法 / 184 5.4.3 处理父请求被重新激活后的回调方法 / 185 5.4.4 启动subrequest子请求 / 185 5.5 subrequest执行过程中的主要场景 / 186 5.5.1 如何启动subrequest / 186 5.5.2 如何转发多个子请求的响应包体 / 188 5.5.3 子请求如何激活父请求 / 192 5.6 subrequest使用的例子 / 193 5.6.1 配置文件中子请求的设置 / 194 5.6.2 请求上下文 / 194 5.6.3 子请求结束时的处理方法 / 195 5.6.4 父请求的回调方法 / 196 5.6.5 启动subrequest / 197 5.7 小结 / 198 第6章 开发一个简单的HTTP过滤模块 / 199 6.1 过滤模块的意义 / 199 6.2 过滤模块的调用顺序 / 200 6.2.1 过滤链表是如何构成的 / 200 6.2.2 过滤链表的顺序 / 203 6.2.3 官方默认HTTP过滤模块的功能简介 / 204 6.3 HTTP过滤模块的开发步骤 / 206 6.4 HTTP过滤模块的简单例子 / 207 6.4.1 如何编写config文件 / 208 6.4.2 配置项和上下文 / 208 6.4.3 定义HTTP过滤模块 / 210 6.4.4 初

<<深入理解Nginx>>

始化HTTP过滤模块 / 211 6.4.5 处理请求中的HTTP头部 / 212 6.4.6 处理请求中的HTTP包体 / 213 6.5 小结 / 214 第7章 Nginx提供的高级数据结构 / 215 7.1 Nginx提供的高级数据结构概述 / 215 7.2 ngx_queue_t双向链表 / 217 7.2.1 为什么设计ngx_queue_t双向链表 / 217 7.2.2 双向链表的使用方法 / 217 7.2.3 使用双向链表排序的例子 / 219 7.2.4 双向链表是如何实现的 / 221 7.3 ngx_array_t动态数组 / 222 7.3.1 为什么设计ngx_array_t动态数组 / 223 7.3.2 动态数组的使用方法 / 223 7.3.3 使用动态数组的例子 / 225 7.3.4 动态数组的扩容方式 / 226 7.4 ngx_list_t单向链表 / 226 7.5 ngx_rbtree_t红黑树 / 227 7.5.1 为什么设计ngx_rbtree_t红黑树 / 227 7.5.2 红黑树的特性 / 228 7.5.3 红黑树的使用方法 / 230 7.5.4 使用红黑树的简单例子 / 233 7.5.5 如何自定义添加成员方法 / 234 7.6 ngx_radix_tree_t基数树 / 236 7.6.1 ngx_radix_tree_t基数树的原理 / 236 7.6.2 基数树的使用方法 / 238 7.6.3 使用基数树的例子 / 239 7.7 支持通配符的散列表 / 240 7.7.1 ngx_hash_t基本散列表 / 240 7.7.2 支持通配符的散列表 / 243 7.7.3 带通配符散列表的使用例子 / 250 7.8 小结 / 254 第三部分 深入Nginx 第8章 Nginx基础架构 / 256 8.1 Web服务器设计中的关键约束 / 256 8.2 Nginx的架构设计 / 259 8.2.1 优秀的模块化设计 / 259 8.2.2 事件驱动架构 / 263 8.2.3 请求的多阶段异步处理 / 264 8.2.4 管理进程、多工作进程设计 / 267 8.2.5 平台无关的代码实现 / 268 8.2.6 内存池的设计 / 268 8.2.7 使用统一管道过滤器模式的HTTP过滤模块 / 268 8.2.8 其他一些用户模块 / 269 8.3 Nginx框架中的核心结构体ngx_cycle_t / 269 8.3.1 ngx_listening_t结构体 / 269 8.3.2 ngx_cycle_t结构体 / 271 8.3.3 ngx_cycle_t支持的方法 / 273 8.4 Nginx启动时框架的处理流程 / 275 8.5 worker进程是如何工作的 / 278 8.6 master进程是如何工作的 / 281 8.7 小结 / 286 第9章 事件模块 / 287 9.1 事件处理框架概述 / 287 9.2 Nginx事件的定义 / 290 9.3 Nginx连接的定义 / 293 9.3.1 被动连接 / 294 9.3.2 主动连接 / 297 9.3.3 ngx_connection_t连接池 / 298 9.4 ngx_events_module核心模块 / 300 9.4.1 如何管理所有事件模块的配置项 / 301 9.4.2 管理事件模块 / 303 9.5 ngx_event_core_module事件模块 / 305 9.6 epoll事件驱动模块 / 310 9.6.1 epoll的原理和用法 / 311 9.6.2 如何使用epoll / 313 9.6.3 ngx_epoll_module模块的实现 / 315 9.7 定时器事件 / 323 9.7.1 缓存时间的管理 / 324 9.7.2 缓存时间的精度 / 326 9.7.3 定时器的实现 / 327 9.8 事件驱动框架的处理流程 / 328 9.8.1 如何建立新连接 / 329 9.8.2 如何解决“惊群”问题 / 330 9.8.3 如何实现负载均衡 / 333 9.8.4 post事件队列 / 334 9.8.5 ngx_process_events_and_timers流程 / 335 9.9 文件的异步I/O / 338 9.9.1 Linux内核提供的文件异步I/O / 339 9.9.2 ngx_epoll_module模块中实现的针对文件的异步I/O / 342 9.10 小结 / 346 第10章 HTTP框架的初始化 / 347 10.1 HTTP框架概述 / 348 10.2 管理HTTP模块的配置项 / 351 10.2.1 管理main级别下的配置项 / 352 10.2.2 管理server级别下的配置项 / 354 10.2.3 管理location级别下的配置项 / 357 10.2.4 不同级别配置项的合并 / 362 10.3 监听端口的管理 / 367 10.4 server的快速检索 / 369 10.5 location的快速检索 / 371 10.6 HTTP请求的11个处理阶段 / 372 10.6.1 HTTP处理阶段的普适规则 / 374 10.6.2 NGX_HTTP_POST_READ_PHASE阶段 / 376 10.6.3 NGX_HTTP_SERVER_REWRITE_PHASE阶段 / 378 10.6.4 NGX_HTTP_FIND_CONFIG_PHASE阶段 / 379 10.6.5 NGX_HTTP_REWRITE_PHASE阶段 / 379 10.6.6 NGX_HTTP_POST_REWRITE_PHASE阶段 / 379 10.6.7 NGX_HTTP_PREACCESS_PHASE阶段 / 379 10.6.8 NGX_HTTP_ACCESS_PHASE阶段 / 380 10.6.9 NGX_HTTP_POST_ACCESS_PHASE阶段 / 380 10.6.10 NGX_HTTP_TRY_FILES_PHASE阶段 / 381 10.6.11 NGX_HTTP_CONTENT_PHASE阶段 / 381 10.6.12 NGX_HTTP_LOG_PHASE阶段 / 382 10.7 HTTP框架的初始化流程 / 383 10.8 小结 / 385 第11章 HTTP框架的执行流程 / 386 11.1 HTTP框架执行流程概述 / 387 11.2 新连接建立时的行为 / 388 11.3 第一次可读事件的处理 / 390 11.4 接收HTTP请求行 / 396 11.5 接收HTTP头部 / 399 11.6 处理HTTP请求 / 403 11.6.1 ngx_http_core_generic_phase / 409 11.6.2 ngx_http_core_rewrite_phase / 411 11.6.3 ngx_http_core_access_phase / 412 11.6.4 ngx_http_core_content_phase / 415 11.7 subrequest与post请求 / 419 11.8 处理HTTP包体 / 421 11.8.1 接收包体 / 422 11.8.2 放弃接收包体 / 429 11.9 发送HTTP响应 / 433 11.9.1 ngx_http_send_header / 434 11.9.2 ngx_http_output_filter / 436 11.9.3 ngx_http_writer / 440 11.10 结束HTTP请求 / 442 11.10.1 ngx_http_close_connection / 443 11.10.2 ngx_http_free_request / 444 11.10.3 ngx_http_close_request / 446 11.10.4 ngx_http_finalize_connection / 447 11.10.5 ngx_http_terminate_request / 447 11.10.6 ngx_http_finalize_request / 448 11.11 小结 / 452 第12章 upstream机制的设计与实现 / 453 12.1 upstream机制概述 / 453 12.1.1 设计目的 / 454 12.1.2 ngx_http_upstream_t数据结构的意义 / 456 12.1.3 ngx_http_upstream_conf_t配置结构体 / 459 12.2 启动upstream / 462 12.3 与上游服务器建立连接 / 464 12.4

<<深入理解Nginx>>

发送请求到上游服务器 / 467 12.5 接收上游服务器的响应头部 / 470 12.5.1 应用层协议的两段划分方式 / 470 12.5.2 处理包体的3种方式 / 471 12.5.3 接收响应头部的流程 / 473 12.6 不转发响应时的处理流程 / 476 12.6.1 input_filter方法的设计 / 477 12.6.2 默认的input_filter方法 / 478 12.6.3 接收包体的流程 / 479 12.7 以下游网速优先来转发响应 / 481 12.7.1 转发响应的包头 / 482 12.7.2 转发响应的包体 / 484 12.8 以上游网速优先来转发响应 / 489 12.8.1 ngx_event_pipe_t结构体的意义 / 489 12.8.2 转发响应的包头 / 493 12.8.3 转发响应的包体 / 495 12.8.4 ngx_event_pipe_read_upstream方法 / 498 12.8.5 ngx_event_pipe_write_to_downstream方法 / 502 12.9 结束upstream请求 / 504 12.10 小结 / 508 第13章 邮件代理模块 / 509 13.1 邮件代理服务器的功能 / 509 13.2 邮件模块的处理框架 / 512 13.2.1 一个请求的8个独立处理阶段 / 512 13.2.2 邮件类模块的定义 / 514 13.2.3 邮件框架的初始化 / 516 13.3 初始化请求 / 517 13.3.1 描述邮件请求的ngx_mail_session_t结构体 / 517 13.3.2 初始化邮件请求的流程 / 519 13.4 接收并解析客户端请求 / 520 13.5 邮件认证 / 520 13.5.1 ngx_mail_auth_http_ctx_t结构体 / 520 13.5.2 与认证服务器建立连接 / 522 13.5.3 发送请求到认证服务器 / 522 13.5.4 接收并解析响应 / 525 13.6 与上游邮件服务器间的认证交互 / 526 13.6.1 ngx_mail_proxy_ctx_t结构体 / 526 13.6.2 向上游邮件服务器发起连接 / 527 13.6.3 与邮件服务器认证交互的过程 / 528 13.7 透传上游邮件服务器与客户端间的流 / 530 13.8 小结 / 535 第14章 进程间的通信机制 / 536 14.1 概述 / 536 14.2 共享内存 / 536 14.3 原子操作 / 541 14.3.1 不支持原子库下的原子操作 / 541 14.3.2 x86架构下的原子操作 / 542 14.3.3 自旋锁 / 545 14.4 Nginx频道 / 546 14.5 信号 / 549 14.6 信号量 / 551 14.7 文件锁 / 553 14.8 互斥锁 / 556 14.8.1 文件锁实现的ngx_shmtx_t锁 / 558 14.8.2 原子变量实现的ngx_shmtx_t锁 / 560 14.9 小结 / 565

章节摘录

版权页：插图：3) 如果handler方法返回NGX_DONE，则意味着刚才的handler方法无法在这一次调度中处理完这一个阶段，它需要多次的调度才能完成。

注意，此时返回NGX_OK，它会使得HTTP框架立刻把控制权交还给epoll等事件模块，不再处理当前请求，唯有这个请求上的事件再次被触发时才会继续执行。

4) 如果handler方法返回除去NGX_DECLINED或者NGX_DONE以外的其他值，则调用ngx_http_finalize_request结束请求，其参数为handler方法的返回值。

可以注意到，ngx_http_core_rewrite_phase方法与ngx_http_core_generic_phase方法有一个显著的不同点：前者永远不会导致跨过同一个HTTP阶段的其他处理方法，就直接跳到下一个阶段来处理请求。

原因其实很简单，可能有许多HTTP模块在NGX_HTTP_SERVERREWRITE_PHASE和NGX_HTTP_REWRITE_PHASE阶段同时处理重写URL这样的业务，HTTP框架认为这两个阶段的HTTP模块是完全平等的，序号靠前的HTTP模块优先级并不会更高，它不能决定序号靠后的HTTP模块是否可以再次重写URL。

因此，ngx_http_core_rewrite_phase方法绝对不会把phase_handler直接设置到下一个阶段处理方法的流程中，即不可能存在类似下面的代码。

11.6.3 ngx_http_core_access_phase ngx_http_core_access_phase方法是仅用于NGX_HTTP_ACCESS_PHASE阶段的处理方法，这一阶段用于控制用户发起的请求是否合法，如检测客户端的IP地址是否允许访问

。它涉及nginx.conf配置文件中satisfy配置项的参数值，见表11—2。

对于表11—2的any配置项，是通过ngx_http_request_t结构体中的access_code成员来传递handler方法的返回值的，因此，ngx_http_core_access_phase方法会比较复杂，如图11—10所示。

<<深入理解Nginx>>

编辑推荐

《深入理解Nginx:模块开发与架构解析》由阿里巴巴资深Nginx专家撰写，透彻解析Nginx架构，详解Nginx模块开发方法和技巧。

<<深入理解Nginx>>

名人推荐

从耐心帮助读者了解“如何阅读本书”到书中详细的代码解析与注释、大量而精致的各种图表，以及为本书开发的在线支持网站等，无不体现了作者全身心投入本书的写作。

更为可贵的是，作者在实际工作中一行行阅读Nginx源代码、不断尝试和探索，耗费了巨大的精力所积累的宝贵经验，无私地和读者分享。

这也成就了Nginx这方面的巨作——覆盖了Nginx的安装、配置、模块开发、架构解析和深度应用等各个方面，适合不同层次的读者，并能切实地帮助读者有效地解决Nginx应用中所碰到的困惑与难题。

——朱少民 同济大学软件学院教授这是国内（或许也是国外）第一本关于Nginx模块开发的书籍。

作者有着丰富的Nginx开发和运维经验，其定制的Nginx服务于真实的大并发SaaS应用，因此其作品不是一本泛泛的手册，而是经验教训之心血结晶。

这本书的面世，对于致力于打造符合自己应用场景之高性能Web服务器的开发人员来说，无疑是一大福音。

——Grant pan, 思科CRDC Senior Manager在互联网上，关于如何安装及配置Nginx的文章可谓众多，可惜一直以来却缺少面向开发人员对其架构原理及核心模块进行系统阐述的相关著作。

本书面向不同层次的读者，对Nginx的使用、配置、架构原理及模块开发进行了系统而细致的阐述，无论是单纯使用Nginx的系统工程师还是专注于高性能服务器端研发的开发人员，都可以在本书中发现你所需的内容。

向Igor Sysoev致敬!感谢陶辉为我们带来这本很棒的书!

——范昕 思科（美国），Senior EngineerNginx是一个功能丰富、插件（模块）众多、性能卓越的Web服务器，业界多把它放在业务的最前端充当静态资源服务器或者反向代理服务器，应用广泛。

本书循序渐进地揭开了Nginx的面纱，从如何使用原生的Nginx入手，进而以几个简明的例子为主线说明如何开发http模块，最后再综合介绍Nginx的完整设计思路，帮助读者快速、深入地掌握如何基于Nginx开发出高性能服务器。

——吴峥涛 阿里巴巴云计算公司 架构师

<<深入理解Nginx>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>