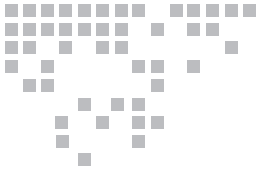
第一部分 Part 1

Nginx 操作基础



Chapter 1 第 1 章

Nginx 高效服务器

Nginx 是一款轻量级的 Web 服务器，特点是高性能、高并发。它由俄罗斯程序设计师 Igor Sysoev 开发，供俄国大型入口网站及搜索引擎 Rambler 使用。Nginx 在 BSD-like 协议下发行，是一款高性能 Web 服务器，目前在 Web 服务器中排名第二。虽然 Apache 还是全球 Web 服务器的“老大”，但是 Nginx 已经占到了 Wed 服务器市场 22% 以上的份额，是成长最快的 Web 服务器。Nginx 使用了大量的高并发和低内存占用技术，并使用了高可靠性技术，拥有高过 Apache 一个数量级以上的接入能力。因为并发能力强的特点，Nginx 在中国的互联网公司中得到了大量应用，中国的大型互联网公司无一不使用了 Nginx，以应对中国众多的网民，以及各种抢购热潮（如“双十一”）、世界杯等热点事件。Nginx 在这种大量的流量涌入、需要分流、导流、反向代理的场合下得到大量应用。

1.1 Nginx 的特点

与其他 Web 服务器相比，Nginx 具有以下显著特点。

1. 速度更快

Nginx 使用了预读、连接池、内存池等技术，使得单次 HTTP 请求速度更快。同时，因为其整体的多进程架构以及轻任务思想，在更多连接的情况下（以万为单位的并发情况下），Nginx 比其他 Web 服务器速度更快。

2. 扩展性好

Nginx 的结构是“核心 + 模块”的结构，Nginx 本身就是一个基于 Epoll 或 Kqueue 的事件处理和分发架构，管理 HTTP 主流程，其他功能都可以通过模块实现。模块专注于自

身功能实现，可以更稳定，模块的升级和修复不影响其他功能以及核心本身。模块可以不断添加或升级，如事件（event）模块、代理模块、过滤模块、请求地址获取模块、地址转换模块、应答处理模块、日志模块等。Nginx 提供了众多的模块以供选择，可以配置出不同行为的 Web 服务器。

Nginx 提供了 C 级别的模块开发机制，但 C 级别的开发需要遵从复杂的数据结构。现在可以通过 ngx_lua 模块以 Lua 脚本实现业务逻辑。得益于 Lua 协程的支持，ngx_lua 在万级并发请求时只占用很少的内存，而性能都是万级（Operation Per Second，每秒操作次数），这使得 Nginx 的扩展性更好。

3. 高可靠性

得益于整体架构的优秀以及模块设计的简单性，Nginx 拥有极高的可靠性，在各大型网站中得到了认可。Nginx 核心由一个任务很轻的管理进程（master 进程）和若干工作进程（worker 进程）组成。具体的 HTTP 请求在工作进程内负载均衡，如果某个工作进程异常终止了，管理进程会迅速重启一个新的工作进程接替该进程。

4. 低内存占用

一般情况下，10000 个非活跃的 HTTP 保活连接仅占用 2.5MB 内存。而 ngx_lua 每扩展 10000 个连接也仅占 2.xMB 内存，使得 Nginx 可以大量部署。

5. 高并发能力

一般 Nginx 是部署在万级以上的场合下。为了应付海量的请求，各网站都需要单机能处理峰值 10 万以上并发请求的 Web 服务器。理论上，Nginx 处理能力的上限仅受内存限制，简单的业务场景下 Nginx 还可以提供更高的处理能力。

Nginx 全异步、非阻塞 I/O 的思想贯穿在核心、模块以及 ngx_lua 模块中，无论是自己实现的模块，还是通过 Lua 实现的脚本代码，都是非阻塞地高速运行。

6. 热部署

因为 Nginx 的管理进程和工作进程是分开设计的，所以可以实现热部署功能，即能在系统不间断的情况下升级可执行程序、更新配置文件、更新日志文件等。

7. 开源

Nginx 遵守相对自由的 BSD 协议。用户可以自由使用 Nginx，还可以自由修改和使用 Nginx 的源码。用户可以在节省大量时间和成本的情况下，得到一个高性能的服务器框架。

1.2　Nginx 的安装

Nginx 有预编译版本、源码，因为我们进行的是 Nginx 配套的开发工作，考虑到功能、性能等原因，源码安装方法更适合本书读者，故这里主要介绍源码安装方法。

另外，有一个 OpenResty 项目将 Nginx 整合进 Lua、LuaJIT 和其他配合组件，形成一个多功能的开发型 Web 服务器套件，因为我们学习 Lua 开发，所以推荐使用 OpenResty，本章介绍的 Nginx 安装方法可供读者自定义系统时参考。

因为典型的互联网应用基本安装在 Linux 上，所以，本文中的安装和编译均以 Linux 为例，系统为 CentOS，用其他 Linux 发布包将对应工具换下即可。

1. 下载 Nginx 源文件

安装文件与源码可以从 Nginx 官网下载，地址为 http://nginx.org/en/download.html。

我们需要下载源码，本书以 1.10.0 版本为例。

下载：

```
wget http://nginx.org/download/nginx-1.10.0.tar.gz
```

解压：

```
tar -zxvf nginx-1.10.0.tar.gz
```

2. 检查安装依赖项

要使用 Nginx 常用功能，首先需要确保操作系统上至少安装如下软件。

1）GCC（GNU Compiler Collection）：用来编译 C 语言程序。在使用源码方式安装 Nginx 时，需要使用 GCC 编译 Nginx 及后面要用到的模块源码。

2）PCRE（Perl Compatible Regular Expressions，Perl 兼容正则表达式）：由 Philip Hazel 开发的函数库，目前为很多软件使用，支持正则表达式，由 RegEx 发展而来。ngx_lua 中的 ngx.re.* 系列 API 需要使用 PCRE 库。如果在 nginx.conf 中使用了正则表达式，那么编译 Nginx 时就必须把 PCRE 库编译进 Nginx，Nginx 中的 HTTP 模块要靠它解析正则表达式。通常情况下都会用到正则表达式。

3）Zlib 库：用于对 HTTP 报文内容做 gzip 格式压缩，如果在 nginx.conf 中配置了 gzip on，并指定某些 Content-Type 的 HTTP 应答包体使用 gzip 进行压缩，以减小网络传输量，那么就需要在编译 Nginx 时指定 zlib，将其编译进 Nginx。

4）OpenSSL 库：使用 HTTPS 在 SSL 上传输 HTTP，就需要安装 OpenSSL。另外，在 ngx_lua 中使用 MD5、SHA1 等散列函数时，也需要安装 OpenSSL。

Nginx 的特点是高性能和定制灵活，实现多功能的 Web 服务，所以各种模块可以根据需求组合。这些模块可能使用到其他支撑的基础库，因此需要保证这些库的正确安装，上面列出的 4 个库就是基础库，如果用到了其他库，也需要首先安装，具体需求参见模块的相关文档和手册。

在 Nginx 上安装工具的方法很多，我们可以使用 yum 安装，相对比较简单，命令如下：

```
yum -y install gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel
```

也可以分步骤安装，命令如下：

```
yum install –y gcc
yum install –y gcc-c++
yum install –y pcre pcre-devel
yum install –y openssl openssl-devel
```

3. Linux 内核参数优化

默认的 Linux 内核参数适合于通用的场景。Linux 的特点是可以根据不同的应用场景调整内核参数以及安装的包和工具，使之成为专用的服务器。通常 Linux 用作各种服务器更为合适，作为高性能的 Web 服务器的基础服务是比较典型的应用。

当 Nginx 作为静态 Web 服务器、反向代理服务器等不同服务器时，所需的内核参数是不同的，本文针对通常的多并发 Nginx 应用，给出内核参数修改样表。

修改 /etc/sysctl.conf，常用配置如下：

```
fs.file-max = 999999
net.core.netdev_max_backlog = 8096
net.core.rmem_default = 262144
net.core.wmem_default = 262144
net.core.rmem_max = 2097152
net.core.wmem_max = 2097152
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_keepalive_time = 600
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_max_tw_buckets = 5000
net.ipv4.ip_local_port_range = 1024 61000
net.ipv4.tcp_rmem = 4096 32768 262142
net.ipv4.tcp_wmen = 4096 32768 262142
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 1024
```

执行 sysctl –p 命令，使修改生效。

各内核参数的作用如下。

1）fs.file-max：表示进程（在 Nginx 里指一个工作进程）可以同时打开的最大句柄数。本参数影响最大并发连接数。

2）net.ipv4.tcp_syncookies：解决 TCP 的 SYN 攻击。

3）net.ipv4.tcp_tw_reuse：参数为 1 时表示允许将 TIME_WAIT 状态的套接字重新用于新的 TCP 连接。服务器上的 TCP 协议栈在工作时会有大量的 TIME_WAIT 状态连接，重新使用这些连接对于服务器处理大并发连接非常有用。

4）net.ipv4.tcp_keepalive_time：表示当 keepalive 启用时，TCP 发送 keepalive 消息的频率。默认为 2 小时，如果本值变小，可以更快地清理无效的连接。

5）net.ipv4.tcp_fine_timeout：表示服务器主动关闭连接时，套接字的 FIN_WAIT-2 状态最大时间。

6）net.ipv4.tcp_max_tw_buckets：表示操作系统允许的 TIME_WAIT 套接字数量的最大值。当超过这个值，TIME_WAIT 状态的套接字被立即清除并输出警告消息，默认值为 180000，过多的 TIME_WAIT 套接字会使服务器速度变慢。

7）net.ipv4.tcp_max_syn_backlog：表示 TCP 三次握手阶段 SYN 请求队列最大值，默认为 1024。调置为更大的值可以使 Nginx 在非常繁忙的情况下，若来不及接收新的连接时，Linux 不至于丢失客户端新创建的连接请求。

8）net.ipv4.ip_local_port_range：定义 UDP 和 TCP 连接中本地端口范围（不包括连接到远端的端口）。

9）net.ipv4.tcp_rmen：定义 TCP 接收缓存（TCP 接收窗口）的最小值、默认值、最大值。

10）net.ipv4.tcp_wmen：定义 TCP 发送缓存（TCP 发送窗口）的最小值、默认值、最大值。

11）net.core.netdev_max_backlog：当网卡接收报文速度大于内核处理速度时，本参数设置这个缓冲队列最大值。

12）net.core.rmem_default：表示内核套接字接收缓冲区默认值。

13）net.core.wmem_default：表示内核套接字发送缓冲区默认值。

14）net.core.rmem_max：表示内核套接字接收缓冲区最大值。

15）net.core.wmem_max：表示内核套接字发送缓冲区最大值。

4. 配置安装选项

通常 Nginx 安装在 /opt 或 /usr/local 目录下。其他配置选项根据需要选择，具体意义和使用方法参见 1.3 节中各个配置项分类表格，或使用 ./configuration –help 命令查看。对于具体的 ngx_lua API 使用中所需要的参数，也可以在 API 章节查看 API 描述。

配置命令如下：

```
./configure  –prefix=/opt/nginx –sbin-path=/opt/nginx/sbin/nginx
```

5. 编译与安装

命令如下：

```
make
make install
```

未在 ./configuration 中指定目录的情况下，Nginx 默认会安装到 /usr/local/nginx 目录下。安装后的 Nginx 可以通过复制创建新实例，以方便调试和开发。

6. 启动、停止、重启

启动命令如下：

```
/opt/nginx/sbin/nginx –p /opt/nginx/
```

如果在编译的时候通过 --prefix 和 --sbin-path 指定了目录，可以直接使用 Nginx 启动。

-p 指定 Nginx 的目录。因为同一台服务器可以运行多个 Nginx 实例，所以需要指定当前实例的目录。这个参数将影响很多 ngx_lua API 中文件参数的检索。通常将配置文件等均放在指定目录下面。

多个 Nginx 实例可以编译不同的参数和模块，以实现不同的应用。

在本机浏览器中输入 http://127.0.0.1 或在其他机器上输入 Nginx 所在服务器 IP，如果看到了“Welcome to nginx”页面，表示 nginx 启动成功。

停止命令如下：

```
/opt/nginx/sbin/nginx –p /opt/nginx –s stop
```

如果编译时指定了 Nginx 的工作目录，可以直接使用 nginx –s stop 启动。

重启（重新载入配置文件）命令如下：

```
/opt/nginx/sbin/nginx –p /opt/nginx –s reload
```

如果编译时指定了目录，可以直接使用 nginx –s reload 重启。

1.3　configure 命令参数

使用 configure 命令参数可以在新编译的 Nginx 程序里打包指定的模块，或去除指定的模块，这样可以自定义 Nginx 功能，同时可以减少内存占用。下面介绍常用的 configure 命令参数。

1. 编译参数

编译器相关参数如表 1-1 所示。

表 1-1　编译器相关参数

参数	说明
--with-cc=PATH	C 编译器路径
--with-cpp=PATH	C++ 编译器路径
--with-cc-opt=OPTIONS	链接时使用到第三方库时，需要指定链接参数。如需要使用 hello 库，则可做以下设置：--with-cc-opt=hello –L/usr/local/hello
--with-cpu-opt=CPU	指定 CPU 处理器架构，取值范围为 pentium、pentiumpro、pentium3、pentium4、athlon、opteron、sparc32、sparc、ppc64

2. 路径参数

configures 支持的路径参数如表 1-2 所示。

表 1-2　configure 支持的路径参数

参数	说明	默认值
--prefix=PATH	Nginx 安装部署的根目录	默认为 /usr/local/nginx 目录。可以通过命令行添加 -p 参数动态修改
--sbin-path=PATH	可执行文件放置路径	<prefix>/sbin/nginx 如果设置 --prefix=/usr/local/nginx，则 Nginx 运行文件在 /usr/local/nginx/sbin/nginx
--conf-path=PATH	配置文件存放路径	<prefix>/conf/nginx.conf

（续）

参数	说明	默认值
--error-log-path=PATH	error 错误日志路径，是 error.log 文件路径。可以在 nginx.conf 中配置成不同请求的日志，保存到不同的 log 文件中	<prefix>/log/error.log
--pid-path=PATH	pid 文件存放路径。pid 文件存放以 ASCII 码存放的 Nginx 管理进程 ID。这个 ID 在通过命令行向进程发送命令时使用，而不是通过使用 /usr/local/nginx/sbin/nginx –p /usr/local/nginx –s stop 停止 Nginx	<prefix>/logs/nginx.pid 可以在 nginx.conf 中修改 pid 的目录和文件名。 假设 pid 文件路径为 /data/logs/nginx.pid，则命令如下： kill –QUIT ‘cat /data/logs/nginx.pid’
--lock-path=PATH	lock 文件存放路径	<prefix>/logs/nginx.lock
--builddir=DIR	configure 执行与编译期间产生的临时文件存放目录	<nginx source path>/objs
--with-per_modules_path	perl 模块路径。指定第三方 perl 模块	无
--with-perl=PATH	perl 二进制文件路径。如果配置的 Nginx 要执行 perl 脚本，则需要配置此目录	无
--http-log-path=PATH	access 日志目录，每一个 HTTP 请求处理结束都会记录此日志	<prefix>/log/access.log
--http-proxy-temp-path=PATH	Nginx 作为 HTTP 反向代理时，上游服务器产生的 HTTP 包体需要临时存放在磁盘文件时，将保存在本配置项下的目录中	<prefix>/proxy_temp 一般为了性能，通常将此目录设置为内存虚拟磁盘
--http-fastcgi-temp-path=PATH	FastCGI 使用的临时文件目录	<prefix>/fastcgi_temp
--http-uwsgi-temp-path=PATH	uWSGI 使用的临时文件目录	<prefix>/uwsgi_temp
--http-scgi-temp-path=PATH	SCGI 使用的临时文件目录	<prefix>/scgi_temp

3. 依赖参数

依赖参数如表 1-3 所示。

表 1-3 依赖参数

参数	说明
--without-pcre	如果确认不使用正则表达式，那么使用本参数取消正则表达式支持
--with-pcre	强制使用 PCRE 库
--with-pcre = DIR	指定 PCRE 库源码路径，编译 Nginx 时会进入该目录编译 PCRE 源码
--with-pcre-opt=OPTIONS	编译 PCRE 源码时需要加入的编译选项
--with=libatomic	强制使用 atomic 库。atomic 库是 CPU 架构独立的一种原子操作，支持下列体系：x86（包括 i386 和 x86_64）、PPC64、Sparc64（V9 或更高版本）或者安装了 GCC4.1.0 及以上版本
--with-libatomic=DIR	atomic 库所在的位置
--with-zlib=DIR	指定 zlib 库源码目录，编译 Nginx 时会自动进入该目录编译源码

（续）

参数	说明
--with-zlib-opt=OPTIONS	编译 zlib 源码时希望加入的编译选项
--with-zlib-asm=CPU	指定汇编优化功能，目前支持两种架构：pentium 和 pentiumpro
--with-MD5=DIR	指定 MD5 库源码位置，编译 Nginx 时会进入该目录编译 MD5 源码。注意，Nginx 源码中已经实现了 MD5 算法，如果没有特殊要求，不要指定外在的 MD5 库
--with-MD5-opt=OPTIONS	编译 MD5 时需要加入的选项
--with-MD5-asm	使用 MD5 的汇编源码
--with-SHA1=DIR	指定 SHA1 库源码位置，编译时会进入目录自动编译。注意，OpenSSL 中包含了 SHA1 算法，如果安装了 OpenSSL，那么可以使用 OpenSSL 的算法
--with-SHA1-opt=OPTIONS	编译 SHA1 源码时希望加入的编译选项
--with-SHA1-asm	使用 SHA1 的汇编源码

4. 模块参数

Nginx 的架构分为核心代码（Nginx core）和功能模块（module），以模块形式实现灵活而强大的功能。模块根据需求灵活使用，需要在 configure 阶段把需要用到的模块加载到 Nginx 中来。

Nginx 的模块大致可以分为核心事件（event）模块、默认会编译进 Nginx 的 HTTP 模块、默认不会编译进 Nginx 的 HTTP 模块和其他模块。

Nginx 的核心事件模块相关参数如表 1-4 所示。

表 1-4　Nginx 的核心事件模块相关参数

参数	说明
--with-rtsig_module	使用 rtsig 模块处理事件驱动
--with-select_module	使用 select 模块处理事件驱动。select 是常用的多路复用机制。默认情况下不安装 select 模块，除非找不到其他更好的模块
--withouot-select_module	不安装 select_module
--with-poll_module	使用 poll 模块处理事件驱动 poll 性能与 select 差不多，远不如 epoll 模块。默认情况下不安装 poll 模块
--with-aio_module	使用 AIO 方式处理事件驱动 注意，AIO 只能与 FreeBSD 上的 kqueue 事件处理机制合作，Linux 上无法使用

默认会编译进 Nginx 的 HTTP 模块如表 1-5 所示。

表 1-5　默认会编译进 Nginx 的 HTTP 模块

参数	说明
--without-http_charset_module	不安装 http_charset_module。本模块可以将服务器发出的 HTTP 响应重新编码
--without-http_gzip_module	不安装 http_gzip_module。本模块可以按照配置文件指定的 Content-Type 对特定大小的 HTTP 响应包体进行 gzip 压缩
--without-http_ssi_module	不安装 http_ssi_module。本模块可以在向客户端返回的 HTTP 包体中加入特定的内容，如加入 HTML 文件中固定的页头和页尾

（续）

参数	说明
--without-http_userid_module	不安装 http_userid_module。本模块可以通过 HTTP 请求头部中一些字段认证用户信息，确定请求是否合法
--without-http_access_module	不安装 http_access_module。本模块可以根据 IP 地址限制客户端对服务器的访问
--whout-http_auth_basic_module	不安装 http_auth_basic_module。本模块可以提供最简单的用户名 / 密码认证
--without-http_autoindex_module	不安装 http_autoindex_module。本模块提供简单的目录浏览功能
--without-http_geo_module	不安装 http_geo_module。本模块定义一些常量，用于与客户端 IP 地址关联，Nginx 针对不同地区客户端返回不一样的结果。例如，不同地区显示不同语言的网页
--without-http_map_module	不安装 http_map_module。本模块可以建立一个 key/value 映射表，不同的 key 得到相应的 value，这样可以针对不同的 URL 做特殊处理。例如，返回 302 重定向响应时，可以根据 URL 不同返回不同的 location
--without-http_split_client_module	不安装 http_split_client_module。本模块根据客户端信息如 IP、header 头、cookie 等进行区分处理
--without-http_referer_module	不安装 http_referrer_module。本模块可以根据请求中的 referer 字段拒绝请求
--without-http_rewrite_module	不安装 http_rewrite_module。本模块提供 HTTP 请求的重定向功能，依赖 PCRE 库
--without-http_proxy_module	不安装 http_proxy_module。本模块提供基本的 HTTP 反向代理功能
--without-http_fastcgi_module	不支持 http_fastcgi_module。本模块提供 FastCGI 功能
--without-http_memcached_module	不安装 http_memcached_module。本模块可以直接从 Memcached 服务器中读取数据并返回给客户端
--without-http_limit_zone_module	不安装 http_limit_zone_module。本模块限制某个 IP 地址并发连接数。例如，对一个 IP 限制只允许一个连接
--withou-http_limit_req_module	不安装 http_limit_req_module。本模块限制某个 IP 地址并发请求数
--without-http_empty_gif_module	不安装 http_empty_gif_module。本模块使 Nginx 收到无效请求时，返回内存 1 × 1 像素的 GIF 图片，可以节省系统资源
--without-http_browser_module	不安装 http_browser_module。本模块根据 HTTP 请求中的 user-agent 字段识别浏览器
--without-http_upstream_ip_hash_module	不安装 http_upstream_ip_hash_module，本模块用于在 Nginx 与后端服务器连接时，根据 IP 做散列运算决定与哪台服务器通信，实现可以保持会话的负载均衡

默认不会编译进 Nginx 的 HTTP 模块如表 1-6 所示。

表 1-6　默认不会编译进 Nginx 中的 HTTP 模块

模块	说明
--with-http_ssl_module	安装 http_ssl_module。让 Nginx 支持 SSL 协议，实现 HTTPS 连接
--with-http_realip_module	安装 http_realip_module。本模块从客户端请求的头域信息（如 X-Real-IP 或 X-Forwarded-For）获取真正的客户端 IP

（续）

模块	说明
--with-http_addition_module	安装 http_addition_module。在返回给客户端的 HTTP 包头或包体尾部追加内容
--with-http_image_filter_module	安装 http_image_filter_module。将符合配置的图片实时压缩为指定大小的缩略图再发送给用户。支持 JPEG、PNG、GIF 格式。注意，本模块依赖于 ligd 库，需要在系统内首先安装 ligd
--with-http_xslt_module	安装 http_xslt_modult。可以在将 XML 格式数据发送给客户端之前加入 XSL 渲染
--with-http_sub_module	安装 http_sub_module。可以将返回给客户端的 HTTP 应答包中指定的字符串替换为需要的字符串
--with-http_geoip_module	安装 http_geoip_module。可以依据 MaxMind GeoIP 的 IP 地址数据库分析得到客户端的实际物理地址。本模块依赖于 MaxMind GeoIP 库文件
--with-http_flv_module	安装 http_flv_module。可以在向客户端发送 HTTP 响应时，对 FLV 格式视频文件头部做一些处理，使客户端可以正常观看、拖动
--with-http_dav_module	安装 http_dav_module。让 Nginx 支持 Webdav 标准，如支持 Webdav 中的 put、delete、copy、mkcol、move 等请求
--with-http_gzip_static_module	安装 http_gzip_static_module。可以在做 gzip 压缩前首先检查是否已经存在经过 gzip 压缩的 gz 文件，如果有则直接返回。提前在服务器上压缩好文档，以减少压缩带来的系统开销
--with-http_mp4_module	安装 http_mp4_module。使客户端可以观看、拖动 MP4 视频
--with-http_random_index_module	安装 http_random_index_module。在客户端访问某个目录时，随机返回该目录下的任意文件
--with-http_secure_link_module	安装 http_secure_link_module。提供一种验证请求是否有效的机制。例如，验证 URL 中需要加入的 token 参数是否属于特定客户端
--with-http_sub_status_module	安装 http_sub_status_module。提供 Nginx 性能统计页面
--with-http_degradation_module	安装 http_degradation_module。本模块针对一些特殊的系统调用做优化，但暂时不支持 Linux 系统
--with-google_perftools_module	安装 google_perftools_module。提供 Google 性能测试工具

其他 Configure 参数如表 1-7 所示。

表 1-7　其他 configure 参数

参数	说明
--user=USER	声明 Nginx 工作进程运行时所属用户。不要将工作进程所属用户设成 root。工作进程用户级别低于管理进程用户以便于管理，在工作进程意外中止时，管理进程可以正常启动工作进程
--group=GROUP	设置工作进程运行时所属的组
--with-debug	将 debug 级别的日志编译进 Nginx。也可以在配置文件中修改日志级别
--without-http	禁用 HTTP 服务器
--without-http-cache	禁用 HTTP 服务器里的缓存 Cache 特性
--with-ipv6	支持 IPV6
--with-file-aio	启用文件异步 I/O 功能处理磁盘文件，需要 Linux 支持异步 I/O

（续）

参数	说明
--add-module=PATH	在 Nginx 里加入第三方模块时，指定第三方模块的路径
--with-mail	安装邮件服务器反向代理模块，使 Nginx 可以反射代理 IMAP、POP3、SMTP 等协议
--with-mail_ssl_module	安装 mail_ssl_module。基于 SSL/TLS 协议使用邮件，依赖于 OpenSSL 库
--without-mail_pop3_module	不安装 mail_pop3_module。使用 --with-mail 参数后，本模块默认安装
--without-mail_imap_module	不安装 mail_imap_module。使用 --with-mail 参数后，本模块默认安装
--without-mail_smtp_module	不安装 mail_smtp_module。使用 --with-mail 参数后，本模块默认安装
--with-stream_core_module	安装 ngx_stream_core_module。从 1.9.0 开始加入，支持 TCP 处理能力或负载均衡，默认不加入，使用本参数打开

1.4 小结

本章介绍了 Nginx 服务器的特点、以源码方式编译和安装 Nginx 的方法、Nginx 常用操作，并分类别介绍了 configure 命令参数。