# **Nginx技术点整理**

# **[1. Nginx的优点](https://segmentfault.com/a/1190000020812206)**

Nginx的特点

I/O 多路复用：https://www.zhihu.com/question/28594409/answer/52835876#showWechatShareTip

轻量级

CPU亲和：可以把每个worker进程固定在一个CPU上运行、避免频繁地切换CPU

Sendfile：可以高效的处理静态资源文件的访问

内置服务器 VS Nginx

例如express框架会监听3000端口，spring-boot框架会监听8080端口，能否将这些端口直接对外提供服务，而不使用nginx呢？

答案就是最好不要，nginx还具有如下功能：

动静分离、缓存数据、缓冲请求和响应，提升性能

负载均衡，提高产品的可用性

安全，可以将可疑IP直接加入黑名单

灵活配置监听的域名和端口，将请求按照不同路由进行分发

作者：思过崖

链接：https://segmentfault.com/a/1190000020812206

来源：SegmentFault 思否

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

# **[2. Nginx的安装与开机自启](https://segmentfault.com/a/1190000020839997)**IMG_256

### **下载**

官方下载地址：[http://nginx.org/en/download....](http://nginx.org/en/download.html" \t "https://segmentfault.com/a/_blank)

mkdir -p /usr/local/src

wget http://nginx.org/download/nginx-1.16.1.tar.gz

tar -xvf nginx-1.16.1.tar.gz

cd nginx-1.16.1

### **安装**

安装依赖

yum install -y libxslt-devel.x86\_64 gd.x86\_64 gd-devel.x86\_64 perl-devel perl-ExtUtils-Embed GeoIP.x86\_64 GeoIP-devel.x86\_64 gperftools.x86\_64 gperftools-devel.x86\_64 pcre-devel.x86\_64 openssl-devel.x86\_64

编译配置

./configure --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --modules-path=/usr/lib64/nginx/modules --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --http-client-body-temp-path=/var/cache/nginx/client\_temp --http-proxy-temp-path=/var/cache/nginx/proxy\_temp --http-fastcgi-temp-path=/var/cache/nginx/fastcgi\_temp --http-uwsgi-temp-path=/var/cache/nginx/uwsgi\_temp --http-scgi-temp-path=/var/cache/nginx/scgi\_temp --user=nginx --group=nginx --with-compat --with-file-aio --with-threads --with-http\_addition\_module --with-http\_auth\_request\_module --with-http\_dav\_module --with-http\_flv\_module --with-http\_gunzip\_module --with-http\_gzip\_static\_module --with-http\_mp4\_module --with-http\_random\_index\_module --with-http\_realip\_module --with-http\_secure\_link\_module --with-http\_slice\_module --with-http\_ssl\_module --with-http\_stub\_status\_module --with-http\_sub\_module --with-http\_v2\_module --with-mail --with-mail\_ssl\_module --with-stream --with-stream\_realip\_module --with-stream\_ssl\_module --with-stream\_ssl\_preread\_module --with-debug --with-http\_geoip\_module --with-cc-opt='-O2 -g -pipe -Wall -Wp,-D\_FORTIFY\_SOURCE=2 -fexceptions -fstack-protector-strong --param=ssp-buffer-size=4 -grecord-gcc-switches -m64 -mtune=generic -fPIC' --with-ld-opt='-Wl,-z,relro -Wl,-z,now -pie'

make

make install

安装完成之后，我们可以通过如下命令，获取当前安装 Nginx 时设定的编译配置。

/usr/sbin/nginx -V

依据这个信息，我们以后可以将编译的 Nginx 进行平缓升级。

### **开机自启**

依据 Nginx 官方文档上的说明，[https://www.nginx.com/resources/wiki/start/topics/examples/systemd/](https://www.nginx.com/resources/wiki/start/topics/examples/systemd/" \t "https://segmentfault.com/a/_blank)

**[Unit]**Description=The NGINX HTTP and reverse proxy serverAfter=syslog.target network.target remote-fs.target nss-lookup.target

**[Service]**Type=forkingPIDFile=/run/nginx.pidExecStartPre=/usr/sbin/nginx -tExecStart=/usr/sbin/nginxExecReload=/usr/sbin/nginx -s reloadExecStop=/bin/kill -s QUIT $MAINPIDPrivateTmp=true

**[Install]**WantedBy=multi-user.target

我们只需要将以上内容保存到 /lib/systemd/system/nginx.service，然后 systemctl enable nginx 即可

# **[3. Nginx目录和配置语法](https://segmentfault.com/a/1190000020840122)**IMG_256

### **全局指令**

| **指令** | **值** |
| --- | --- |
| user | 出于安全考虑，默认是nginx、nobody |
| worker\_processes | 工作进程数，一般来说，设置与CPU的核心数相同即可 |
| error\_log | 保存错误日志的路径，可以设置error\_log的级别 |
| pid | nginx 进程id |

user nginx;worker\_processes 4;error\_log /var/log/nginx/error.log warn;pid /var/run/nginx.pid;**events** {

*# 每秒处理多少个客户端连接*

worker\_connections 1024;

}**http** {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

*#与客户端保持连接的最长时间*

keepalive\_timeout 65;

gzip on;

include /etc/nginx/conf.d/\*.conf;

}

### **日志分析**

* ngxtop 流量实时监测 [https://github.com/lebinh/ngxtop](https://github.com/lebinh/ngxtop" \t "https://segmentfault.com/a/_blank)，这个简直就是神器(o^^o)
* goaccess 日志可视化 [https://github.com/allinurl/g...](https://github.com/allinurl/goaccess" \t "https://segmentfault.com/a/_blank)
* goaccess 分析配置 [https://github.com/stockrt/ng...](https://github.com/stockrt/nginx2goaccess" \t "https://segmentfault.com/a/_blank)
* goaccess 操作说明 [https://www.fanhaobai.com/201...](https://www.fanhaobai.com/2017/06/go-access.html" \t "https://segmentfault.com/a/_blank)

### **日志分割**

日积月累，nginx 的日志文件也会变得越来越大，如果我们是自己编译安装的nginx，可能就需要自己来处理日志文件的分割了。定时分割日志文件，这有利于降低单个文件的大小，方便排查，同时只保留最近一段时间的日志，也可以节省磁盘空间

一般采用 logrotate 来进行日志文件的分割：[https://linux.die.net/man/8/l...](https://linux.die.net/man/8/logrotate" \t "https://segmentfault.com/a/_blank)

*# USR1 re-opening log files http://nginx.org/en/docs/control.html# daily 每天处理一次# missingok 忽略文件不存在的错误# rotate 设置旧文件保留的数量# compress 对日志进行压缩# delaycompress 延迟压缩# notifempty 如果文件内容为空，不处理# create 设置文件权限、用户、组# sharedscripts 对于prerotate和postrotate脚本，如果匹配到了多个日志文件，只运行一次脚本。如果没有匹配到，则不运行。# postrotate表示在日志rotate之后，执行的脚本*

**/var/log/nginx/\*.log** {

daily

missingok

rotate 30

compress

delaycompress

notifempty

create 640 nginx nginx

sharedscripts

postrotate

if [ -f /var/run/nginx.pid ]; then

kill -USR1 `cat /var/run/nginx.pid`

fi

endscript}

测试 logrotate 配置是否生效

logrotate -v /etc/logrotate.conf 调试定时任务，不会处理实际文件  
logrotate -vf /etc/logrotate.conf 立刻执行所有任务，不管是否已到执行时间，会处理实际文件

### **server配置**

**server** {

listen 80;

server\_name localhost;

*#nginx程序默认的web根目录*

location / {

root /usr/share/nginx/html;

index index.html index.htm;

}

error\_page 404 /404.html;

*# 重定向服务器的错误页到指定的静态页面 /50x.html*

error\_page 500 502 503 504 404 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

location ~ \.php$ {

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

}

}

当我们配置好了Nginx，就可以使用curl来进行初步的测试了

curl -v http://127.0.0.1/ > /dev/null

倘若我们想禁止用户通过IP或者未设置的域名来访问，可以采取如下措施

**server** {

listen 80 default;

rewrite ^(.\*) http://www.example.com permanent;

}

### **变量**

| **变量类型** | **变量列表** |
| --- | --- |
| 内置变量 | [http://nginx.org/en/docs/vari...](http://nginx.org/en/docs/varindex.html" \t "https://segmentfault.com/a/_blank) |
| HTTP请求变量 | [http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_core_module.html" \l "var_status" \t "https://segmentfault.com/a/_blank) |

参考文档：[http://www.ttlsa.com/nginx/ng...](http://www.ttlsa.com/nginx/nginx-var-1/" \t "https://segmentfault.com/a/_blank)

### **路由**

如果我们想要访问/path/to/name，那么 Nginx 配置中，路由匹配的优先级如下

| **修饰符** | **说明** | **优先级** |
| --- | --- | --- |
| location = /path/to/name | 精确匹配 | 1 |
| location /path/to/name | 完整匹配 | 2 |
| location ^~ /path/to/name | 表示uri以某个常规字符串开头，非正则表达式 | 3 |
| location ~\* /path/to/name | 使用不区分大小写的正则来进行匹配 | 4 |
| location ~ /path/to/name | 使用区分大小写的正则来进行匹配 | 4 |
| location /path | 部分匹配 | 5 |
| location / | 通用匹配 | 6 |

# **[4. Nginx模块](https://segmentfault.com/a/1190000022183871)**

### **Nginx官方模块**

#### **1.ngx\_http\_stub\_status\_module**

[http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_stub_status_module.html" \t "https://segmentfault.com/a/_blank)。

此模块可以查看nginx对数据包处理的基本信息

*#启用方法*location /status {

stub\_status;

}

访问 /status，展示的数据如下

Active connections: 4

server accepts handled requests

310840 310840 481035

Reading: 0 Writing: 1 Waiting: 3

#### **2. ngx\_http\_random\_index\_module**

[http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_random_index_module.html" \t "https://segmentfault.com/a/_blank)

此模块用于在目录下，随机地取用某个文件，作为默认主页。假如 /usr/share/nginx/html/random 目录下有 aaa.html、bbb.html、ccc.html 三个文件，则使用如下配置即可

location /random {

root /usr/share/nginx/html;

random\_index on;

}

如下代码会随机返回 aaa、bbb、ccc 这三个html之一

curl http://127.0.0.1/random/

#### **3. ngx\_http\_sub\_module**

[http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_sub_module.html" \t "https://segmentfault.com/a/_blank)

用于替换掉响应内容中的指定字符串。

location / {

sub\_filter '<a href="https://www.baidu.com/' '<a href="https://www.qq.com/';

*#如果文件中有多处需要替换，只替换第一处*

sub\_filter\_once on;

*#保留替换前，原始的最后修改时间*

sub\_filter\_last\_modified on;

*#默认只替换 text/html 这一MIME类型*

sub\_filter\_types text/html;

}

#### **4. ngx\_http\_limit\_conn\_module**

[http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_limit_req_module.html" \t "https://segmentfault.com/a/_blank)

| **HTTP协议** | **请求与连接** | **说明** |
| --- | --- | --- |
| HTTP 1.0 | TCP不能复用 | 一个连接，一个请求 |
| HTTP 1.1 | 顺序性TCP复用 | 一个连接，可以按顺序的发出多个请求 |
| HTTP 2.0 | 多路TCP复用 | 一个连接，可以并行的发出多个请求 |

我们可以把nginx的zone，理解成一个动态的数据库，1M的zone内存可以存储至少16000条记录，判断用户请求是否合法，就是不断查询当前IP在数据库中记录的数量是否超出了限制。

在如下的路由请求中，我们限制用户的并发连接数分别为5个和10个。

注意：****并发是指同一时刻的请求量，与每秒的请求数量是有区别的。****

**http** {

limit\_conn\_zone $binary\_remote\_addr zone=addr\_conn:10m;

...

**server** {

...

location / {

limit\_conn addr\_conn 5;

...

}

location ~ \.php$ {

limit\_conn addr\_conn 10;

...

}

}

}

为了测试代码效果，我们可以安装 httpd-tools 进行ab测试。  
对于limit\_conn 的测试，不管是内网测试，还是外网测试，都是可行的。

yum install httpd-tools

ab -n 20 -c 20 http://127.0.0.1/

执行 ab -n 20 -c 20 http://127.0.0.1/，总共20个连接，失败了15个，成功建立的连接数只有5个了

| **条目** | **数值** |
| --- | --- |
| Complete requests: | 20 |
| Failed requests: | 15 |

执行 ab -n 20 -c 20 http://127.0.0.1/index.php，总共20个连接，失败了10个，成功建立的连接数确实只有10个

| **条目** | **数值** |
| --- | --- |
| Complete requests: | 20 |
| Failed requests: | 10 |

经过以上测试，可以得出的结论就是：****代码是不会骗人的****

对于频率限制造成的 Failed requests ，我们在 nginx 的 error.log 中也能发现错误记录。如果错误日志过多，我们就需要排查一下访问量是否正常。如果正常的话，又需要考虑一下如何优化参数设置。

2017/07/31 11:41:37 [error] 24550#0: \*580766 limiting connections by zone "addr\_conn", client: 119.130.188.64, server: www.siguoya.name, request: "GET / HTTP/1.0", host: "www.siguoya.name"

2017/07/31 11:41:37 [error] 24550#0: \*580767 limiting connections by zone "addr\_conn", client: 119.130.188.64, server: www.siguoya.name, request: "GET / HTTP/1.0", host: "www.siguoya.name"

#### **5.ngx\_http\_limit\_req\_module**

[http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_limit_req_module.html" \t "https://segmentfault.com/a/_blank)

ngx\_http\_limit\_conn\_module 用于限制并发连接数，而 ngx\_http\_limit\_req\_module 则用于限制并发请求数。实际业务场景中，通常使用 ngx\_http\_limit\_req\_module 会更多一些。

**http** {

limit\_req\_zone $binary\_remote\_addr zone=addr\_req:10m rate=1r/s;

...

**server** {

...

location / {

limit\_req zone=addr\_req;

}

}

}

#### **6.ngx\_http\_access\_module**

[http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_access_module.html" \t "https://segmentfault.com/a/_blank)

该模块可以通过限制 ip 来实现访问控制，但弊端在于我们有时无法获取到真实的IP，而且用户的IP也是可以动态变化的，另外还比较容易出现误杀的情况

可以通过以下方法来弥补：

* x\_forwarded\_for，但是并不可靠，存在被修改或者未传递的可能
* geo 模块
* 通过HTTP自定义变量来传递

location / {

allow 192.168.1.0/24;

deny all;

}

#### **7.ngx\_http\_auth\_basic\_module**

[http://nginx.org/en/docs/http...](http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html" \t "https://segmentfault.com/a/_blank)

该模块可以要求用户在访问特定页面的时候，必须输入正确的账号和密码才可以访问，从而实现一个比较初级的权限控制功能

location /auth\_basic {

auth\_basic "custom comment for this auth\_basic";

auth\_basic\_user\_file /usr/local/nginx/conf/.passwd;

}

关于 auth\_basic\_user\_file 使用到的文件，我们可以使用 htpasswd 这个工具来实现

**#**新建一个授权文件

htpasswd -bc username password >> .passwd**#**在已有的授权文件中新加一个用户

htpasswd -b username password >> .passwd

ngx\_http\_auth\_basic\_module 的缺陷如下：用户权限依赖于授权文件，容易造成企业有多个用户账号体系，运维麻烦。

解决方法：

* 结合 lua 实现高效认证
* 利用 nginx-auth-ldap 模块，和 ldap 打通

### **Nginx第三方模块**

有兴趣的可以访问 [https://www.nginx.com/resourc...](https://www.nginx.com/resources/wiki/modules/" \t "https://segmentfault.com/a/_blank)，这里就不一一展开介绍了

# **[5. Nginx静态资源处理](https://segmentfault.com/a/1190000022184905)**IMG_256

### **基本配置**

*#开启sendfile*sendfile on; *#sendfile开启的情况下，针对多个tcp请求，进行延时打包处理，可以有效提高网络包的传输效率*tcp\_nopush on;*#keepalive连接下，对tcp请求立即响应，与tcp\_nopush恰好相反，可以提高网络包的传输实时性*tcp\_nodelay off;

*#对响应的内容，进行压缩*gzip on;*#设置压缩等级，范围1～9，数值越大，文件越小。压缩等级太高的话，对服务器和浏览器的性能都有很大影响*gzip\_comp\_level 1;*#表明gzip http协议的版本*gzip\_http\_version 1.1;*#设置压缩类型，默认情况下只会压缩text/html*gzip\_types text/html image/png image/jpeg

### **资源压缩相关模块**

| **模块** | **说明** |
| --- | --- |
| ngx\_http\_gzip\_module | 提供基础的gzip支持 |
| ngx\_http\_gzip\_static\_module | 在向客户端返回常规文件之前，先查找是否存在该常规文件的.gz文件，有的话就直接返回.gz文件 |
| ngx\_http\_gunzip\_module | 如果响应内容是gzip的，而浏览器不支持的话，可以通过此模块在返回给浏览器之前进行解压 |

ngx\_http\_gzip\_module 是默认编译到nginx的发行版本里面的，用户访问的同时，对文件进行压缩，耗CPU。  
ngx\_http\_gzip\_static\_module 需要编译进去才能有。提前压缩，然后用户访问的时候，返回已经压缩好的，耗磁盘空间。

*# always 就是不管浏览器支不支持都返回.gz内容，此时最好开启ngx\_http\_gunzip\_module模块*gzip\_static always;

是否使用gzip，以及使用不同的gzip方式，http的响应头如下：

* 未使用 gzip

Accept-Ranges:bytesContent-Length:75882

* 使用 gzip on; 的方式

Content-Encoding:gzipTransfer-Encoding:chunked

* 使用 gzip\_static on; 的方式

Content-Encoding:gzipContent-Length:65052

# **[6. Nginx浏览器缓存原理](https://segmentfault.com/a/1190000022184927)**IMG_256

### **与缓存有关的请求头**

| **key** | **value** | **说明** |
| --- | --- | --- |
| Cache-Control | max-age=0 | 客户端向服务器端表明，如果缓存过期了，但是过期时间还没有超过指定时间（以秒为单位），则直接返回304即可。 |
| If-Modified-Since | Wed, 09 Aug 2017 10:28:55 GMT | 根据修改时间来判断文件是否修改过。如果没有修改，则使用缓存。 |
| If-None-Match | 598ae3e7-c | 根据hash来判断文件是否修改过，如果没有修改，则使用缓存。 |

### **与缓存有关的响应头**

| **key** | **value** | **说明** |
| --- | --- | --- |
| Expires | - | http1.0协议中定义的，建议使用更强大的Cache-control |
| Cache-control | max-age=86400 | http1.1协议中定义的，提供了对缓存更细致的控制，优先级也高于Expires。max-age=86400表示客户端在24小时内，无须向服务器端发送请求，可以直接使用本地的缓存文件。但是Chrome等浏览器并没有遵守，而是每次都想浏览器发送请求进行校验。 |
| Last-Modified | Wed, 09 Aug 2017 10:28:55 GMT | 此文件的最后修改时间 |
| ETag | 598ae3e7-c | 此文件的最新hash值 |

作者：思过崖

phper 在进阶的时候总会遇到一些问题和瓶颈，业务代码写多了没有方向感，不知道该从那 里入手去提升，对此我整理了php相关多个知识点高级进阶干货需要的可以免费分享给大家  需要可以加下我 qq:1047403189 或者微信：PHPty123获取哦

这些的话也在下面的直播课中讲解过很多期，并且在不断更新中，是结合企业的一些应用场景讲解的，能够帮助学员突破思维或者带着实战，来帮助大家掌握学习的

电脑 腾讯课堂客户端

手机 腾讯课堂APP

网页版直播地址：https://ke.qq.com/course/328509?tuin=6

2020年 7月3日 晚上20-23点

课题：架构专场分析-PHP+Nginx打造动静分离系统架构第一步  
    授课讲师：PHP学院高级架构师--Luke老师  
    课程地址：IMG_256https://ke.qq.com/course/328509  
    课程内容：  
      1、动静分离架构设计准则? 为何要这样设计  
      2、架构如何落地动静分离  
      3、Nginx调优指南解析  
      4、页面静态化与伪静态原理分析  
      5、高可用架构如何搭建部署

主讲老师介绍:

# 六星教育-Luke老师

六星教育php学院金牌讲师，6年项目研发经验，曾任职于奇虎360 虎牙等公司。精通php主流框架、性能优化、接口开发、拥有大型网站设计和开发经验。