

NginX-Webserver

UnFUG

Christian Fischer

28. April 2011

Inhalt

- 1 **Einleitung**
 - Übersicht
 - Statistiken
 - Vor- und Nachteile
- 2 **Performance**
 - .htaccess
 - Architektur
 - Eigene Erfahrung
- 3 **Konfiguration**
 - Config-Files
 - PHP
 - SSL
 - .htaccess / mod_rewrite Rules konvertieren
- 4 **Quellen**

Übersicht

- Ursprünglich von Igor Sysoev für rambler.ru entwickelt
- Start der Entwicklung im Jahre 2002
- Erste Version am 4. Oktober 2004 veröffentlicht
- Aktuellste Version 1.0.0 vom 12. April 2011
- Lizenz in 2 Absätzen, BSD ähnlich
- Einsatzgebiete:
 - Webserver
 - Load Balancer
 - Reverse Proxy
 - E-Mail-Proxy (POP3/IMAP)

Statistiken

- Einsatz auf Webseiten (weltweit in Prozent)
 - 7.50% aller Webseiten
 - 6.12% der 1 Million Top Webseiten
 - 8.23% aller aktiven Seiten
 - Weiterer, stetiger Wachstum der Anteile
- Einsatz auf Seiten wie: WordPress, Golem, Hulu, Github, Ohloh und SourceForge

Nachteile

- kein .htaccess
- kein mod_security
- keine mod_php / mod_fcgi
- Kommunikation zum Backend nur HTTP 1.0
- Module fest einkompiliert
- kein mod_chroot

Vorteile

- kein .htaccess ;-)
- sehr gute Performance
- niedriger Ressourcenverbrauch
- großer Funktionsumfang
- einfache Konfiguration
- Keine Threads, sondern ereignisbasierte (asynchrone) Architektur

.htaccess

- Jeder Ordner im angeforderten Pfad muss durchsucht werden
- Wenn vorhanden wird jede .htaccess Datei gelesen und verarbeitet
- Dies passiert bei jedem Request

.htaccess

Requests [Per Hour]	Nginx FS Stats	Nginx FS Reads	Apache FS Stats	Apache FS Reads	Comment
1	1	1	6	4	1 req
10	10	10	60	40	10 req
3600	3600	3600	21600	14400	1 req/sec
144000	144000	144000	864000	576000	40 req/sec
324000	324000	324000	194400	1296000	90 req/sec
576000	576000	576000	3456000	2304000	160 req/sec

Architektur

- ereignisbasierte (asynchrone) Architektur
- verarbeitet Anfragen in einem einzigen (oder sehr wenigen) Threads
- Dadurch:
 - Fast gleiche Performance unter wenig Load, hohe Performance unter starkem Load

Eigene Erfahrung

- 3 Foren mit zusammen ca. 1.5 Millionen Posts
- ca. 5.000 User, davon immer ca. 50 online
- vServer mit 1.0 GHz, 1 GB RAM
- Apache mit Standard Debian-Config und mod_php:
CPU Last 80 - 100 %, RAM Verbrauch ca. 500 MB
- NginX mit Standard Debian-Config und php_fcgi:
CPU Last ca. 30%, RAM Verbrauch ca. 50-60 MB

Config-Files

- /etc/nginx/fastcgi_params
- /etc/nginx/mime.types
- /etc/nginx/nginx.conf
- /etc/nginx/sites-available/*
- /etc/nginx/sites-enabled/*

Standard-Config

```
server {  
  
    listen      80;  
    #listen     [::]:80 default ipv6only=on;  
  
    server_name www.host.com host.com;  
  
    access_log  /var/log/nginx/access.log;  
    error_log   /var/log/nginx/error.log;  
  
    location / {  
        root    /var/www/;  
        index   index.html index.htm;  
    }  
}
```

Wichtige Optionen

User und Gruppe unter der der Webserver läuft

```
user www-data nogroup;
```

Für `max_clients`:

```
worker_processes 1; #Sollte auf Anzahl der  
                    #CPU- Cores gesetzt werden
```

```
events {  
    worker_connections 1024;  
}
```

`max_clients = worker_processes * worker_connections`

Wichtige Optionen - Fortsetzung

Timeouts

```
client_body_timeout      30;  
client_header_timeout   30;  
keepalive_timeout       5 5;  
send_timeout            30;  
fastcgi_read_timeout    60;
```

Weitere Optionen

Nur GET, HEAD und POST Methoden erlauben:

```
if ($request_method !~ ^(GET|HEAD|POST)$ ) {  
    return 403;  
}
```

Weitere Optionen - Fortsetzung

Nur Host headers der eigenen Domain erlauben:

```
if ($host !~* ^(host.com|www.host.com)$ ) {  
    return 403;  
}
```

Weitere Optionen - Fortsetzung

Verschiedene Referers verbieten:

```
if ( $http_referer ~* (babes|forsale|girl|jewelry|  
love|nudit|organic|poker|porn|sex|teen|click|  
diamond|poweroversoftware|video|webcam|zippo) ) {  
    return 403;  
}
```

Weitere Optionen - Fortsetzung

Umleitung beim Aufruf einer nicht existierenden Seite auf index.html

```
if (!-e $request_filename) {  
    rewrite . /index.html last;  
}
```

Weitere Optionen - Fortsetzung

http://www.domain.com nach http://domain.com umschreiben

```
if ( $host = 'www.domain.com' ) {  
    rewrite ^/(.*)$ http://domain.com/$1  
    permanent;  
}
```

Weitere Optionen - Fortsetzung

Alle Anfragen nach `https://www.domain.com` umleiten

```
if ( $host ~* ^(domain\.com|www\.domain\.com)$ ) {  
    rewrite ^/(.*)$ https://domain.com/$1  
    permanent;  
}
```

Weitere Optionen - Fortsetzung

Bestimmte User Agents verbieten.

```
if ($http_user_agent ~* (Baiduspider|msnbot) ) {  
    return 403;  
}
```

PHP mit fcgi

PHP 5 cgi muss installiert sein (Paket unter Debian: php5_cgi).
Eintrag in die server direktive:

```
location ~ /\.php$ {
    fastcgi_pass    127.0.0.1:9000;
    fastcgi_index   index.php;
    fastcgi_param   SCRIPT_FILENAME
        /var/www$fastcgi_script_name;
    include fastcgi_params;
}
```

und Erweiterung von:

```
index index.html index.htm index.php;
```

PHP mit Apache Backend-Server

Eintrag in die server direktive:

```
location ~ /\.php$ {  
    proxy_pass http://127.0.0.1;  
}
```

und Erweiterung von:

```
index index.html index.htm index.php;
```

[Sicherheit] PHP Überprüfung ob Datei existiert

Eine sehr wichtige Option bei fcgi- Prozess oder Apache Webserver der im Hintergrund auf weitergeleitete .php Dateien vom NginX-Server lauscht ist:

```
if (!-e $request_filename) {  
    rewrite . /index.php last;  
}
```

direkt nach dem

```
location ~ /\.php$ {
```

Block.

[Sicherheit] PHP Überprüfung ob Datei existiert - Fortsetzung

Durch eine fehlende Überprüfung ist es möglich eine .gif Datei mit schadhaftem Code auf den Server hoch zu laden und über:

```
http://domain.com/upload/schadhafte.gif/index.php
```

aufzurufen. NginX würde die .gif Datei an den Backend-Prozess weiterleiten und dieser würde den schadhaften Code in der .gif Datei ausführen.

[Sicherheit] PHP Überprüfung ob Datei existiert - Fortsetzung

Dies funktioniert jedoch nur, wenn der fcgi- Prozess oder der Apache Webserver auf dem gleichen Server läuft. Alternativen:

- `cgi.fix_pathinfo = false` in der `php.ini`
- Deaktivieren aller Uploads

SSL

Eintrag in die `server` direktive zur Aktivierung von SSL:

```
ssl on;  
ssl_certificate /etc/nginx/ssl/server.crt;  
ssl_certificate_key /etc/nginx/ssl/server.key;  
  
ssl_session_cache shared:SSL:10m;  
  
ssl_prefer_server_ciphers on;
```

Wichtig bei PHP mit `fcgi`:

```
fastcgi_param HTTPS on;
```

vor:

```
fastcgi_pass 127.0.0.1:9000;
```

.htaccess / mod_rewrite Rules konvertieren - Beispiel

```
<FilesMatch Settings.php>  
Order allow,deny  
Deny from all  
</FilesMatch>
```

wird zu:

```
location ~ Settings\.php {  
    deny all;  
}
```

.htaccess / mod_rewrite Rules konvertieren - weitere Beispiele

```
RewriteRule ^sitemap.xml$  
/forum/index.php?action=sitemap;xml
```

wird zu:

```
rewrite sitemap.xml  
"/forum/index.php?action=sitemap;xml" last;
```

.htaccess / mod_rewrite Rules konvertieren - weitere Beispiele

```
RewriteRule ^(activate|admin|announce|arcade|  
attachapprove|buddy|calendar|chat)/?$  
./index.php?pretty;action=$1 [L,QSA]
```

wird zu:

```
rewrite ^/(activate|admin|announce|arcade|  
attachapprove|buddy|calendar|chat)/?$  
"/index.php?pretty;action=$1" last;
```

Quellen

- <http://news.netcraft.com/archives/2011/01/12/january-2011-web-server-survey-4.html>
- <http://www.nginx.org/LICENSE>
- <http://wiki.nginx.org/>
- http://nginx.org/en/docs/http/convertिंग_rewrite_rules.html
- <http://wiki.nginx.org/LikeApache-htaccess>
- <http://www.anilcetin.com/convert-apache-htaccess-to-nginx/>