8 ways to tweak NGINX performance on Linux

If you want to use NGINX to its fullest, you need to tinker with its configuration files and set parameters that will optimize server performance.

NGINX is a popular, free and open source web server. The default NGINX configurations are good enough for the web server to work.

However, if you want to use NGINX to the fullest, you need to tinker with its configuration files and set parameters that will optimize server performance. You will find the configuration files in the /etc/nginx directory on your Linux machine.

1. Configure worker processes in NGINX

The NGINX architecture consists of a master process and several worker processes. The master process's job is to evaluate configuration and manage workers. On the other hand, the role of the worker process is to handle incoming requests and create a connection between cclient and server.

The progress value is set to auto by default. This sets the number of worker processes equal to the number of available CPU cores. To know how many CPU cores are in your system, run the following command:

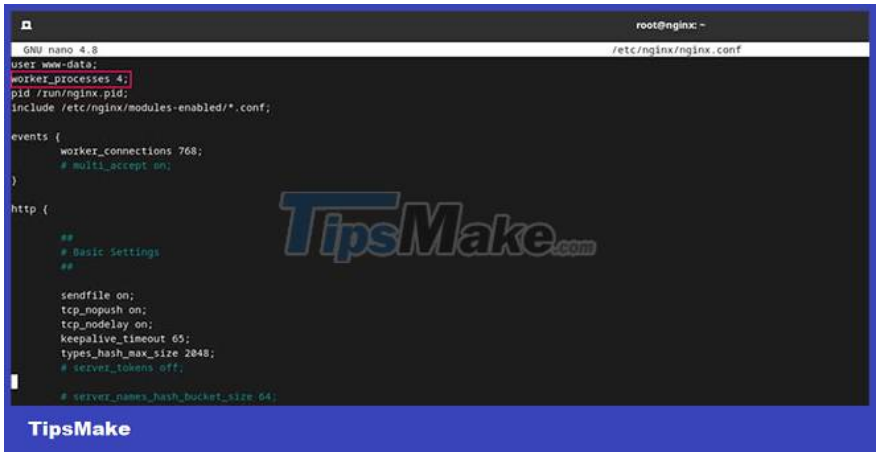
```
grep processor /proc/cpuinfo | wc -l
```

If you want to increase the number of worker processes, you need to do this in the NGINX configuration file.

Open the file with nano:

```
nano etc/nginx/nginx.conf
```

To configure more worker processes, change the default value to the maximum number of available CPU cores in your system.



2. Configure worker connections

Another parameter you can modify to enhance NGINX performance is worker connections. This is the maximum number of TCP connections that each worker process can handle simultaneously.

Most systems have a default value of 512 connections, but many modern systems also support larger numbers. You can check how many connections your system supports with:

```
ulimit -n
```

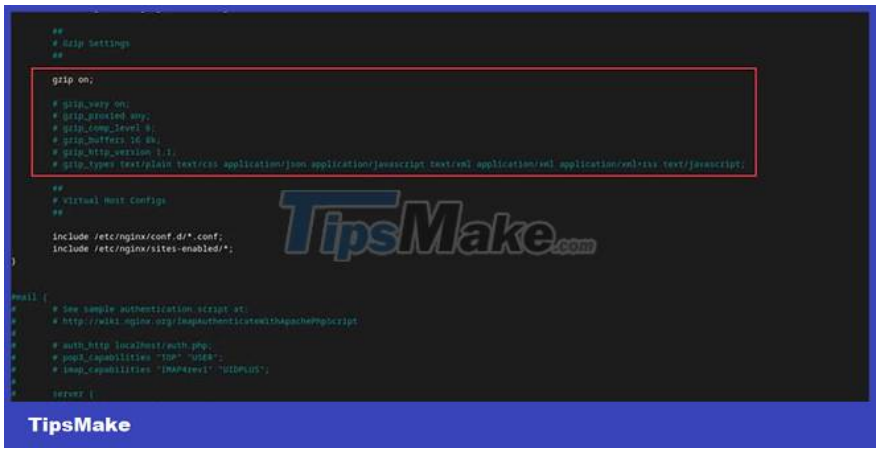
The output will be the maximum number of connections supported. You can then modify the worker_connections variable in the NGINX configuration file to improve performance.

3. Enable GZIP compression in NGINX

NGINX uses GZIP to compress and decompress files. If enabled in the NGINX configuration file, you can save bandwidth and increase website load time on slow connections.

To enable GZIP compression, add the following lines to the NGINX configuration file:

```
server { gzip on; gzip_vary on; gzip_min_length 10240; gzip_proxied expired no-ca
```



4. Limit timeout value in NGINX

Reduced timeout values ??also play an important role in enhancing NGINX performance. Persistent connections reduce the processor and network overhead of opening and closing connections.

You can modify the following parameters in the configuration file to limit timeouts:

```
http { client_body_timeout 12; client_header_timeout 12; keepalive_timeout 15; s
```

5. Adjust buffer size

You can also adjust NGINX buffers to optimize server performance. If the buffer size is too low, NGINX will write to a temporary file causing large I/O operations to run continuously.

You need to set the following buffer parameters for NGINX to work best:

```
http { client_body_buffer_size 10K; client_header_buffer_size 1k; client_max_body
```

6. Disable access logs or enable access log caching

Logs consume large amounts of disk space and CPU/IO cycles which can affect the server's performance if it logs every request.

You can disable access logs, which will save some disk space and CPU processing. To disable access logs, add the following line to the NGINX configuration file:

```
access_log off;
```

Logs are important because they help troubleshoot problems. Disabling logs completely is not a good thing. In this case, you can enable access log caching. This will allow NGINX to cache a batch of logs and put them into the log file at once, instead of applying different log operations to each request.

Add the following line to the NGINX configuration file to enable access log caching:

```
access_log /var/log/nginx/access.log main buffer=16k
```

7. Adjust static content cache time in NGINX

Content on a website that does not change across pages is called static content. Caching of this content allows it to be placed in easily accessible locations. This mechanism reduces bandwidth usage, allows for fast access, and subsequently improves website performance.

When a client requests static content, the server provides a cached version of the content. Add the following lines to the virtual hosts file located in the /etc/nginx/sites-available directory:

```
location ~* .(jpg|jpeg|png|gif|ico|css|js)$ { expires 90d; }
```

This configuration will cache files for 90 days from the last browser access.

8. Enable Open File Cache in NGINX

You can also use the Open File Cache parameters in the NGINX configuration file to enhance its performance. This directive allows file descriptors and frequently accessed files to be cached on the server.

Add the following lines to the http section of the configuration file to enable Open File Cache:

```
http { open_file_cache max=1024 inactive=10s; open_file_cache_valid 60s; open_fi
```

A good practice to follow when changing configurations is to work through each setting one by one and test it. If it works, move on to the next installation. If not, you can always change the configuration back to the default values.

By modifying parameters configured in NGINX configuration files, such as nginx.conf and virtual host files, you can hack NGINX to deliver the best performance.

You finished reading the article "**8 ways to tweak NGINX performance on Linux**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
