Performance Analysis and Optimization of Nginx-based Web Server

Jundi Wang 1*, Zhao Kai 2
1 School of Computer and Artificial Intelligence, Lanzhou Institute of Technology, Lanzhou, Gansu, 730050, China
2 Alumni Office, Lanzhou Institute of Technology, Lanzhou, Gansu, 730050, China
*Corresponding author’s e-mail: wangjd@lzit.edu.cn

Abstract. Performance optimization and analysis of Web server is one of the key technologies of Web server. Through the analysis of the work flow and working principle of the Web server, some influencing factors that are in line with the current environment and have feasibility are selected for targeted optimization, such as value of a thread or process, live time, cache size. Through optimization, the comprehensive performance of Web server based on Nginx has been greatly improved.

1. Introduction
The Web is a distributed graphic information system. Composed of numerous interconnected hypertexts, it is accessible via the Internet. According to the 46th Report on China's Internet Development released by the China Internet Network Information Center in September 2020[1], web applications account for more than 80% of all applications used. In the network environment today, the performance of a website (or system that provides web applications) is often largely decided by its web server. For the maintenance staff of websites, it is necessary to optimize the web server. Recently, many large websites in China begin to use Nginx to deal with high concurrent access, because Nginx has the advantages of lightweight and high concurrency. This paper discusses how to boost the performance of Nginx-based web servers, with the CentOS 7 system employed as the operating system of the Web server.

2. About Web server performance analysis
2.1. Web server’s workflow
Although there is a wide range of Web servers and the implementation details differ across different Web containers, their underlying operating principles are similar, as shown in Fig. 1.

2.2. Definition and evaluation indicators
The main indicators used to measure and evaluate the web server performance include: Transaction response time, Request response time, Number of concurrent users, Throughput, Transaction per second and resource utilization[2].
3. Nginx-based Web server performance optimization

3.1. Affecting factors of Web server

The performance of Web servers mostly depends on the Web container, the operating system, and the hardware[3].

(1) Web container. Web containers can be divided into HTTP Server (Apache and Nginx) and Application Server (Tomcat). The application scenarios and optimization strategies of different web containers often affect the performance of Web servers.

(2) Operating system. Factors that affect the operating system include the number of TCP connections, the number of files opened at the same time, the number of processes running at the same time, the size of the buffer that receives sockets in the kernel, and the size of the cache sent from and received by TCP.

(3) Hardware. For a stand-alone system, there are always bottlenecks, which are often related to factors like the number of CPU cores, frequency, and cache; disk capacity, read/write rate, and cache; memory size, frequency; and the number of network cards and transmission rate.

As discussed above, the main factors that affect the performance of Web servers are the value domain and survival or release time of the Web container threads and processes; the value domain of Linux kernel's TCP sending and receiving buffer, the size of the buffer that sends and receives the kernel sockets, and the connection timeout and survival of sockets; and the number of Web server CPU. The performance optimization shall focus on these aspects as well.

3.2. Optimization methods of Web server

The underlying environment and the Web container are designed, installed and implemented as below.

(1) Linux configuration strategies

- The hardware configuration for VMware includes 1 1-core and 1-thread CPU, 2G memory, 40G hard disk, 1 network card in NAT mode. The CentOS7 system is installed in the Minimal mode.
- Install htop and nmon as the server performance monitoring tool, net-tool toolkit as the network monitoring tool, wget to provide the network download function, and setools-console and policycoreutils-python to manage the configuration of SeLinux.

(2) Related configuration strategies

- Install related tools and services. Configure the firewall of CentOS7 system.
- Install mod_ssl and openssl to support SSL verification, i.e. HTTPS support; use openssl to create a key with self-signed verification; modify corresponding configuration file; and add related configurations for SSL verification.
- Modify the SELinux access permission to the root directory of Web server to meet the httpd_sys_content_t.
3.2.1. Linux optimization
Edit the file of /etc/sysctl.d/99-sysctl.conf and add the configuration strategies in Table 1.

| Item                        | Description                                                                 |
|-----------------------------|-----------------------------------------------------------------------------|
| fs.file-max                 | The maximum number of handles that can be opened by a single process in the Linux operating system is 999999 |
| net.ipv4.tcp_tw_reuse       | Sockets in the TIME_WAIT state are available to new TCP connections          |
| net.ipv4.tcp_fin_timeout    | When the Web server closes the socket connection, the maximum survival time of the socket in the FIN_WAIT_2 state is 10ms |
| net.ipv4.tcp_max_tw_buckets | The maximum sockets in the TIME_WAIT state in the Web server are 100,000     |
| net.ipv4.ip_local_port_range| The available range of local port for UDP and TCP connection is between 1024 and 65000  |
| net.core.netdev_max_backlog| Let the maximum data packets stored in queue in the Linux kernel be 5120 bytes, when the number of data packets received by the server network card is greater than the kernel processing rate |
| net.ipv4.tcp_syncookies     | Disable the SYN cookies authentication to solve the denial of service attack launched against TCP connection |
| net.ipv4.tcp_max_syn_backlog| Let the maximum length of the SYN request queue during the establishment of TCP three-way handshake be 81920 bytes |
| net.ipv4.tcp_tw_recycle     | Set a fast recovery mechanism when time wait is enabled                      |
| net.ipv4.tcp_max_orphans    | Let the maximum TCP sockets associated with any user's file handle be 819,200 |

3.2.2. Nginx container optimization
Edit the /usr/local/nginx/conf/nginx.conf file[4].
- Enable one daemon when Nginx is started by default.
- Let the maximum matching files in the Worker process be 20480.
- Let the maximum number of connections in each process be 20480.
- Let Nginx transmit the static resources in the kernel directly, without reading and writing, thus reducing system overhead and improving resource utilization.
- Let all Nginx data packets sorted in the queue and sent when they grow into specified size, thus reducing network overhead and improving transmission efficiency.
- Disable Nagle so that Nginx can send data quickly to shorten the response time.
- Let the request and response timeout be 5ms, the header buffer size be 8k, and the maximum byte size of a single file be 4k.

4. Web server performance test
Most web applications are distributed today, which makes it impossible for the Web server to figure out the number of requesters. It is therefore necessary to specify the number of concurrent requesters that a Web server can handle. The quality of hardware, the capacity to deal with concurrent requesters, the stability of system, and the responsiveness are all critical to the operation and maintenance of a Web server.

JMeter and Nmon are employed to test the Web server performance: JMeter tests the performance of software and server, and generates performance test reports; and Nmon monitors and analyzes the performance data.
4.1. Test implementation
There are seven pages (with massive text and pictures. The home page contains multimedia information) on the website to be tested. The pages and the website as a whole are tested with JMeter. The number of threads and cycles is set to be 1000 and 10 respectively. In other words, 70,000 concurrent tests are made about this website. Use nmon -s 1-c 120 -f -m ./ to create a 120-second monitoring process on the server monitored. After the monitoring completes, use Nmon and Nmon analyser to analyze the results. Below are the test results.

4.1.1. Performance test results before optimization
(1)Throughput test. The test results on the throughput of Nginx Web server before optimization are shown in Table 2.

| test page      | sample size | throughput | Receive (KB/sec) | Send (KB/sec) |
|----------------|-------------|------------|------------------|---------------|
| home page      | 10000       | 250.1/sec  | 4420.65          | 27.78         |
| test page 1    | 10000       | 250.2/sec  | 3041.70          | 33.21         |
| test page 2    | 10000       | 250.4/sec  | 3278.66          | 32.51         |
| test page 3    | 10000       | 250.6/sec  | 3229.65          | 32.53         |
| test page 4    | 10000       | 250.8/sec  | 3226.80          | 32.57         |
| test page 5    | 10000       | 250.0/sec  | 3299.72          | 34.07         |
| test page 6    | 10000       | 250.2/sec  | 2772.43          | 33.35         |
| total          | 70000       | 1750.2/sec | 23218.11         | 225.49        |

As shown in the table, before optimization, the throughput of the static website deployed on the Nginx server was 1750.2/sec.

(2)Response time. The test results on the response time of the Nginx Web server before optimization are shown in Figure 2.

The response time of each page shows that, before optimization, the average response time of the static website deployed on the Nginx server was 295 milliseconds, 90% of which was 334 milliseconds.
4.1.2. Performance test results after optimization
(1) Throughput test. The test results on the throughput of Nginx Web server after optimization are shown in Table 3.

| test page   | sample size | throughput | Receive (KB/sec) | Send (KB/sec) |
|-------------|-------------|------------|------------------|---------------|
| home page   | 10000       | 376.6/sec  | 6816.81          | 43.03         |
| test page 1 | 10000       | 376.9/sec  | 4581.07          | 50.02         |
| test page 2 | 10000       | 376.9/sec  | 4935.59          | 48.94         |
| test page 3 | 10000       | 377.1/sec  | 4861.04          | 48.97         |
| test page 4 | 10000       | 377.5/sec  | 4857.36          | 49.03         |
| test page 5 | 10000       | 377.8/sec  | 4965.19          | 51.26         |
| test page 6 | 10000       | 378.5/sec  | 4177.94          | 50.27         |
| total       | 70000       | 2635.8/sec | 35126.38         | 340.79        |

As shown in the table, after optimization, the throughput of the static website deployed on the Nginx server was 2635.8/sec, registering a 50.59% increase.

(2) Response time. The test results on the response time of the Nginx Web server after optimization are shown in Figure 3.

The response time of each page shows that, after optimization, the average response time of the static website deployed on the Nginx server was 246 milliseconds (19.91% decrease), 90% of which was 280 milliseconds (19.28% decrease).
4.2. Comparison and analysis of the test results
Based on comparison and calculation of the performance test results of the Web server before and after optimization, data on throughput, response time, and hardware consumption is shown in Table 4.

| test content     | before optimization | after optimization | performance change     |
|------------------|---------------------|--------------------|------------------------|
| throughput       | 1750.2/sec          | 2635.8/sec         | rose by 50.59%         |
| response time    | 295ms               | 246ms              | decreased by 19.91%    |
| 90% response time| 334ms               | 280ms              | decreased by 19.28%    |
| memory available | 0.5GB               | 1.3GB              | rose by 145.45%        |

5. Conclusions
This paper studies several factors that affect the performance of the Web server, such as the value domain and survival or release time of the Web container threads and processes, the value domain of Linux kernel's TCP sending and receiving buffer, the size of the buffer that sends and receives the kernel sockets, and the connection timeout and survival of sockets. The issue of incomplete performance releases of the Web server in the initial installation state is addressed, improving the performance of the Web server. The solution is feasible and universally applicable.

ACKNOWLEDGMENT
This research was financially supported by ‘Kaiwu’ Innovation Team Support Project of Lanzhou Institute of Technology.(2018KW-02).

REFERENCES
[1] CNNIC. The 46th Statistical Report on Internet Development in China [R].BeiJing: China Internet Network Information Center,2020: 12-13.
[2] Yue J, Wen Y, Huang ML. (2019) The implementation of high concurrency of nginx in Windows Environment. Electronic Technology & Software Engineering, 17:47-49.
[3] Wu YB, Liang SHG.(2017) Optimization of the property of web server based on Nginx and Http2.0. Journal of Xinyu University, 22(04):6-8
[4] Cao H, Sun Q.(2017) Intelligent Housekeeping Service System Based on Nginx Server. Industrial Control Computer, 30(04),57-59