FreeBSD + nginx: Best WWW server for the best OS

Sergey A. Osokin
Moscow, Russia
Ports committer
FreeBSD Project
osa@FreeBSD.org

Abstract

Today the NGINX web server can be safely considered mature. Launched 10 years ago the project is still gaining popularity. This paper introduced the NGINX webserver, describes its implementation approach and architectural goals. Also it demonstrates how NGINX works on FreeBSD operating system and reveals strategies of the product usage, the ways to deploy and optimize t and other challenges.

1 Introduction

You probably remember the tagline of one of the most popular email mail user agent: *All mail clients suck. This one just sucks less.* OK. To paraphrase the message to the topic of my speech I'd like to say: *All web servers suck. This one just sucks less.*

Now let's be seriuos.

Today Internet is different from what it was 10-20 years ago. Those day it was a collection of a small number of HTML pages sometimes interconnected by hyperlinks. An there were some emails around. And now Internet has huge search engines, video streams, nancial transactions, voice data.

The total number of devices of any kind connected to Internet is constantly growing and soon it will be bigger than the total Earth population more than two times.

Everything changes and technology never goes back and software changes too. Are you ready for the modern data flows? So, why whould anyone will use old software in new century?

Today more then 10% of the top of 100000 busiest sites run on NGINX, including Groupon, LivingSocial, Zappos, Playdom, WordPress, Yandex, TechCrunch.

2 What is NGINX?

What is NGINX? NGINX is a HTTP server and HTTP/mail proxy server under 2-clause BSD licence.

The basic functions of HTTP server are:

- serving static requests, index files, automatically create list of files;
- accelerated proxying with caching;
- modularity (more original and third-party modules);
- byte-ranges, chunked answers;
- Server Side Includes (SSI);
- SSL (secure socket layer).

Additional functions of HTTP server are:

- virtual servers (by IP and hostname);
- keep-alive support and pipelined connections;
- configuration flexibility (ability to change timeouts and size of buffers);
- ability to ¡¡hot update¿¿ of the main executable file on-the-fly, without any service disruption, the old process is stopped afterward;
- log files customization;
- limit speed rate for answers;
- URI modifications by using regular expression;
- customized error pages for 400 and 500 error codes;
- embedded perl.

Other functions are:

- User redirection to IMAP/POP3 backend using an external HTTP authentication server;
- User authentication through an external HTTP authentication server and connection redirection to an internal SMTP backend;
- Authentication methods:
  - POP3: USER/PASS, APOP, AUTH LOGINPLAINCRAM-MD5;
  - IMAP: LOGIN, AUTH LOGINPLAINCRAM-MD5;
  - SMTP: AUTH LOGINPLAINCRAM-MD5;
- SSL support;
- STARTTLS and STLS support.

3 NGINX architecture and scalability

NGINX supports various operating system, i.e.

- FreeBSD 3, 4 on `i386`; FreeBSD 5 - 10 on `i386` and `amd64`;
- Linux 2.2 - 2.6 on `i386`; Linux 2.6 on `amd64`;
- Solaris 9 on `i386` and `sun4u`; Solaris 10 on `i386`, `amd64`, `sun4v`;
- AIX 7.1 on `powerpc`;

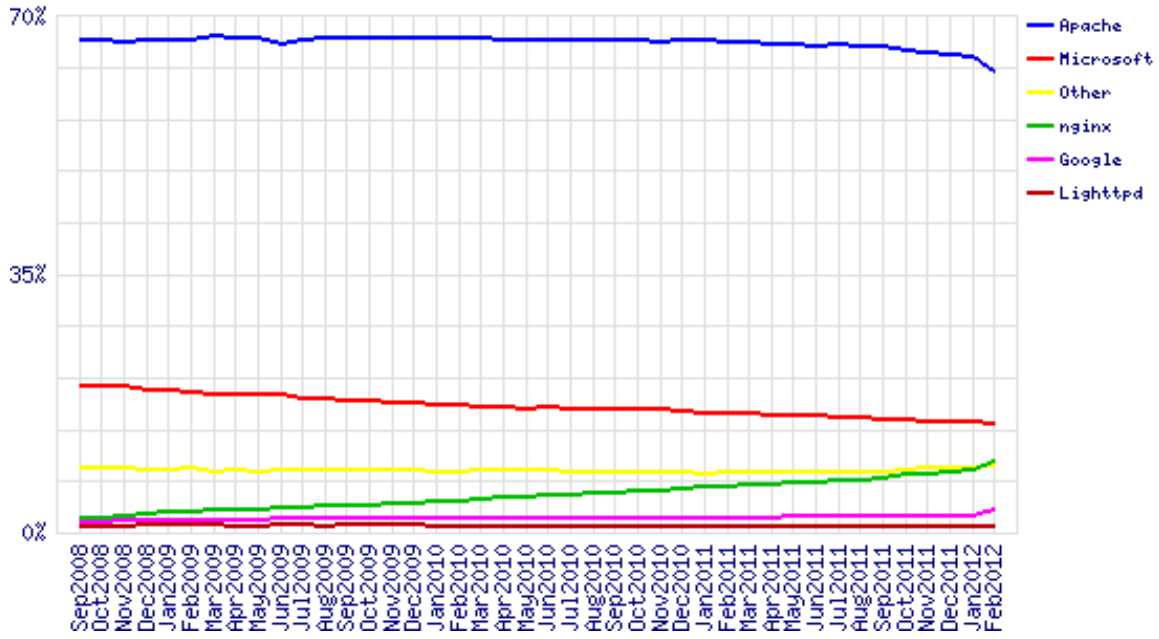

Figure 1: Market Share for Top Servers Across the Million Busiest Sites September 2008 - February 2012, Netcraft.com

- Mac OS X on `ppc`, `i386`;
- Windows XP, Windows Server 2003.

NGINX on FreeBSD uses `kqueue(2)` notification mechanism and supports the various features of it including `EV_CLEAR`, `EV_DISABLE` (to temporarily disable events), `NOTE_LOWAT`, `EV_EOF`, number of available data, error codes.

For other operating system nginx can use different event-base models or toolkits, such as `epoll` for linux, `/dev/pool` for Solaris, HP-UX, and `IO completion ports` for Windows.

Modern operating systems provide similar functions:

- for file transfers, `sendfile` on FreeBSD, linux and Mac OS X, `sendfile64` on linux and `sendfilev` on Solaris are used correspondingly;
- asynchronous operations with file descriptors, `aio_read(2)`/`aio_write(2)`;
- non-cached direct access, `O_DIRECT` on FreeBSD ;
- accept-filters, `accf_http(9)` on FreeBSD, and `TCP_DEFER_ACCEPT` on linux.

10,000 inactive HTTP keep-alive connections take about 2.5M memory. Data copy operations are kept to minimum.

4 NGINX processes

One master and several worker processes; worker processes run under an unprivileged user account.

At the start time NGINX runs both master and worker processes. Worker processes are ready to serve traffic for users.

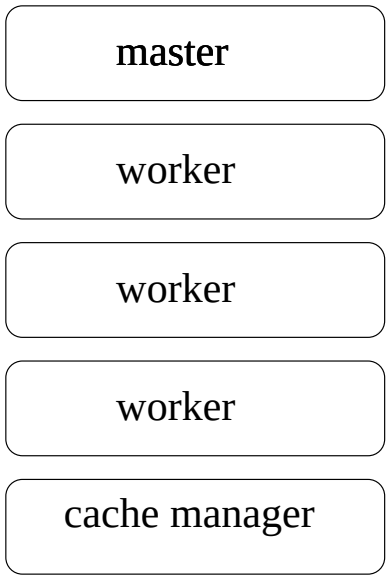

Figure 2: NGINX processes

5 NGINX vs lighttpd

lighttpd has the same event-base model, as nginx. Unfortunately, lighttpd has some long-unresolvable annoying bugs. During the last year, since Jan 2011, the developers of lighttpd made only 9 commits in they svn trunk.

6 NGINX vs apache-2.2.x

apache-2.2.x has different process-based model.

```
accept(5,{ AF_INET 127.0.0.1:31135 },0x7fffffffd648) = 3 (0x3)
kevent(9,{0x3,EVFILT_READ,EV_ADD|EV_ENABLE|EV_CLEAR,0,0x0,0x8035a0150},1,
       {0x3,EVFILT_READ,EV_CLEAR,0,0x10,0x8035a0150},512,{60.000000000 }) = 1 (0x1)
gettimeofday({1329404560.553483 },0x0)  = 0 (0x0)
recvfrom(3,"GET / HTTP/1.0\r\n",1024,0x0,NULL,0x0) = 16 (0x10)
kevent(9,{},0,{0x3,EVFILT_READ,EV_CLEAR,0,0x2,0x8035a0150},
       512,{54.573000000 }) = 1 (0x1)
gettimeofday({1329404561.631475 },0x0)  = 0 (0x0)
recvfrom(3,"\r\n",1008,0x0,NULL,0x0)  = 2 (0x2)
stat("/usr/local/www/nginx/index.html",
     { mode=-rw-r--r-- ,inode=2037452,size=64,blksize=32768 }) = 0 (0x0)
open("/usr/local/www/nginx/index.html",O_NONBLOCK,00) = 10 (0xa)
fstat(10,{ mode=-rw-r--r-- ,inode=2037452,size=64,blksize=32768 }) = 0 (0x0)
pread(0xa,0x80341ef20,0x40,0x0,0x50,0x2)  = 64 (0x40)
sendfile(0xa,0x3,0x0,0x40,0x7fffffffc8c0,0x7fffffffc940) = 0 (0x0)
clock_gettime(13,{1329404561.000000000 })  = 0 (0x0)
sendto(4,"<29>Feb 16 19:02:41 nginx: 127.0"...,101,0x0,NULL,0x0) = 101 (0x65)
close(10)  = 0 (0x0)
close(3)  = 0 (0x0)
```

Figure 3: NGINX worker processing request

7 How to NGINX process request

Let's see how to NGINX process a request. First let's connect to a worker process with `truss(1)` to record system calls, the do the typical `GET / HTTP/1.0` request by `telnet 127.0.0.1 80` from command line.

As you can see the worker NGINX performs a minimum possible number of system calls. Following the `accept(2)` call, it takes request with `recvfrom(2)`, then call `stat(2)` to index file, `open(2)` it and after `fstat(2)` call worker use `sendfile(2)` call for send data.

8 Performance tuning FreeBSD for NGINX

Here is a set of kernel configuration values to get maximum performance of NGINX on FreeBSD operating system running on `amd64` platform.

```
net.inet.tcp.syncache.hashsize=1024
net.inet.tcp.syncache.bucketlimit=100
net.inet.tcp.tcbhashsize=4096
```

Figure 4: Kernel configuration values in `/boot/loader.conf` for FreeBSD on `amd64`

9 Performance benchmarking

I did a performance benchmarking tests with apache-2.2, lighttpd and nginx. All test performs on HP DL160 with Intel©Xeon©E5504 2GHz, 4G memory server with FreeBSD 9.0 for `amd64` onboard.

I used `benchmarks/autobench` from ports tree

```
kern.ipc.maxsockets=204800
kern.ipc.somaxconn=4096
kern.maxfiles=204800
kern.maxfilesperproc=200000
net.inet.ip.portrange.first=1024
net.inet.ip.portrange.last=65535
net.inet.ip.portrange.randomized=0
net.inet.tcp.blackhole=1
net.inet.tcp.fast\_finwait2\_recycle=1
net.inet.tcp.maxtcptw=40960
net.inet.tcp.msl=30000
net.inet.tcp.recvspace=8192
net.inet.tcp.syncookies=1
net.inet.udp.blackhole=1
```

Figure 5: Kernel configuration values in `/etc/sysctl.conf` for FreeBSD on `amd64`

to perform the tests. `autobench` is a Perl script for automating the process of benchmarking a webserver. The script is a wrapper around `httperf`. Autobench runs httperf a number of times against host, increasing the number of requested connections per second on each iteration, and extracts the significant data from the `httperf` output, delivering a TSV format file which I convert to graphs.

All webservers worked fine when I used 10KB and 100KB files for test. But situation dramatically changed when I run test with 1M file against apache-2.2. During a test run on the server load average rose to 35, the number of httpd processes had increased to 1,000. To resolve this problem, I tipped the apache on localhost: 8080 and provided him the role of backend, but as a frontend, I installed nginx. Also, I allowed nginx to cache responses

```
# autobench --single_host --host1 192.168.254.2 --uri1 /${f} --quiet \
   --low_rate 100 --high_rate 1000 --rate_step 100 --num_call 10 \
   --num_conn 5000 --timeout 5 --file ${s}-${f}.tsv
```

Figure 6: Performance benchmarking with `autobench` tool

from the backend.

As result,

The situation has changed for the better: the speed of the network input-output increased, reduced memory consumption, CPU load has decreased.

10 NGINX in FreeBSD ports tree

There are two versions of NGINX available in the FreeBSD ports tree: stable and development. The stable version can be used for production. The development version can be used by developers to work on they own modules. I started supporting NGINX in the ports tree 7 years ago. It used to be a very simple port with an original rewrite module. After 6 years `www/nginx` and `www/nginx-devel` ports supports 35 third-party modules. By default, the port builds a vanilla (vendor) version, with all third-party modules are disabled by default.

11 Acknowledgments

12 Availability

General information on the NGINX web server, as well as releases of NGINX may be found on the NGINX web page `http://www.nginx.org/`.

A lot of information about how to install NGINX, additional and third-party modules

Notes

1. Thus, this is not a footnote

References

[1] *What is NGINX?*, `http://nginx.com/papers/nginx-wp.pdf`

[2] *One-page NGINX description*, `http://nginx.com/papers/nginx-features.pdf`

[3] kqueue(2) man page, The FreeBSD Documentation Project, The FreeBSD Project

[4] truss(1) man page, The FreeBSD Documentation Project, The FreeBSD Project

[5] I/O Completion Ports (Windows), `http://msdn.microsoft.com/en-us/library/aa365198(VS.85).aspx`

[6] February 2012 Web Server Survey, `http://news.netcraft.com/archives/2012/02/07/february-2012-web-server-survey.html`

[7] A secure, fast, compliant, and very flexible Web Server, `http://www.lighttpd.net`

[8] Apache web server, `http://httpd.apache.org`

[9] httpperf — a tool for measuring web server performance, `http://code.google.com/p/httperf/`

[10] Autobench — automating the process of benchmarking, `http://www.xenoclast.org/autobench/`

```
location / {
proxy_pass http://127.0.0.1:8080/;
proxy_redirect off;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

client_max_body_size 40m;
client_body_buffer_size 256k;
proxy_connect_timeout 120;
proxy_send_timeout 120;
proxy_read_timeout 120;
proxy_buffer_size 64k;
proxy_buffers 4 64k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size 64k;

proxy_cache_valid 200 301 302 304 30m;
proxy_cache_key
 "$request_method|$http_if_modified_since|$http_if_none_match|$host|$request_uri";
proxy_hide_header "Set-Cookie";
proxy_ignore_headers "Cache-Control" "Expires";
proxy_cache wholepage;
}
```

Figure 7: The part of `nginx.conf` for proxy request and caching result

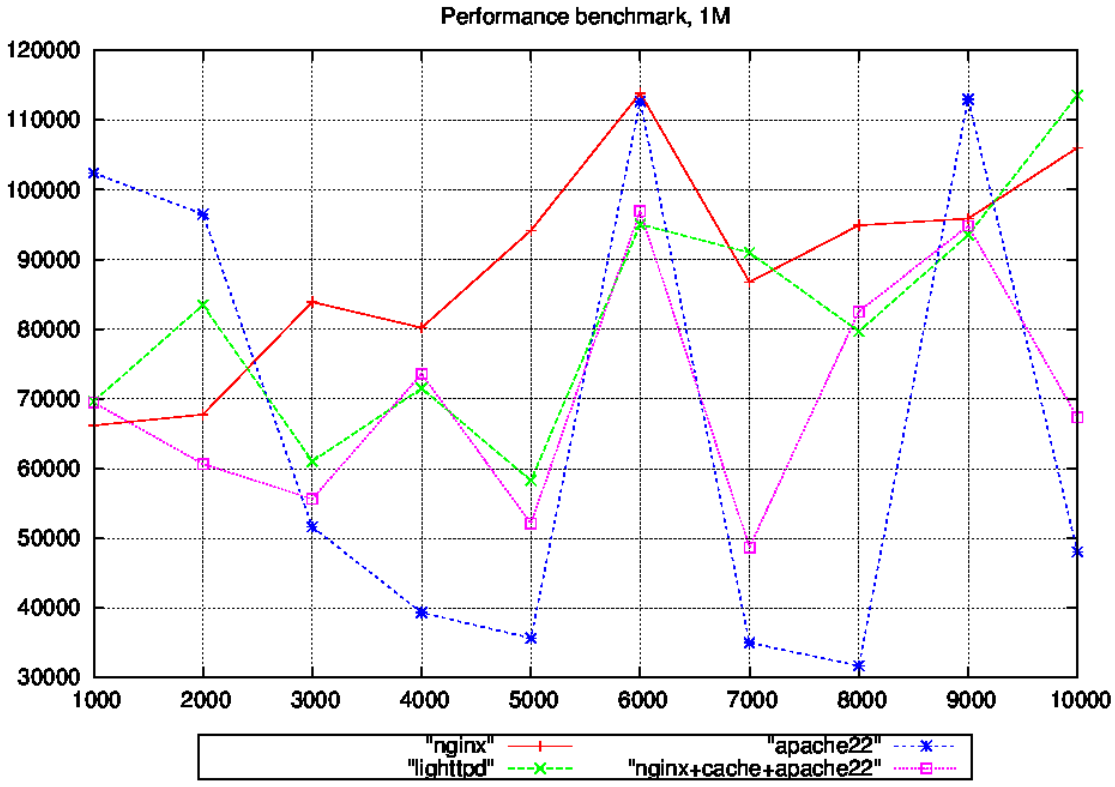

Figure 8: Benchmark results