Table of Contents

The difference between Lua and LuaJIT

Lua is very efficient. It runs faster than many other scripts (such as Perl, Python, and Ruby). This has been confirmed in third-party independent evaluations. In spite of this, there will still be people who are not satisfied. They always feel that "ah, not fast enough!". LuaJIT is an attempt to squeeze out some speed. It uses Just-in Time technology to compile Lua code into machine code and then hand it directly to CPU for execution. The evaluation report of LuaJIT 2 shows that its acceleration effect is significant in many aspects such as numerical operations, loops and function calls, coroutine switching, and string manipulation. With the FFI feature, LuaJIT 2 is also much faster than the standard Lua interpreter in scenarios that require frequent calls to external C/C++ code. LuaJIT 2 currently supports many different architectures including i386, x86_64, ARM, PowerPC, and MIPS.

LuaJIT is a Lua interpreter and just-in-time compiler written in C and assembly language. LuaJIT is designed to be fully compatible with the standard Lua 5.1 language, while optionally supporting some of the useful features in Lua 5.2 and Lua 5.3 that do not undermine backward compatibility. Therefore, the standard Lua language code can be run on LuaJIT without modification. One big difference between LuaJIT and the standard Lua interpreter is that LuaJIT's execution speed, even if it's written in Lua interpreter, is much faster than the standard Lua 5.1 interpreter. It can be said to be a highly efficient Lua implementation. Another difference is that LuaJIT supports more basic primitives and features than the standard Lua 5.1 language, and therefore is more powerful in functionality.

Unless otherwise specified, our next sections are based on LuaJIT.

Lua environment to build

Build on centos 7

Install luajit rpm package: `#yum install luajit`

Verify LUAJIT is installed successfully

```
# luajit -v
LuaJIT 2.1.0-beta1 -- Copyright (C) 2005-2015 Mike Pall.
Http://luajit.org/
```

NGINX

Because Nginx uses an event-driven architecture that can handle millions of TCP connections concurrently, the highly modular design and free licenses make it possible to extend third-party modules that extend Nginx functionality. Therefore, it is widely used as a Web server on high-traffic websites, including large-scale websites such as Taobao, Tencent, Sina, and Jingdong.

Why choose Nginx

Why Nginx? Because it has the following features:

1, processing response request is very fast

In normal circumstances, a single request will get a faster response. At peak times, Nginx can respond to requests faster than other Web servers.

2, high concurrent connection

With the rapid development of the Internet and the continuous increase in the number of Internet users, some large companies and websites need to face high concurrent requests. If there is a server that can withstand more than 100,000 simultaneous requests at the peak, it will surely get everyone's favor. In theory, the upper limit of concurrent connections supported by Nginx depends on your memory, and 100,000 is far from capped.

3, low memory consumption

In normal circumstances, 10,000 inactive HTTP Keep-Alive connections consume only 2.5MB of memory in Nginx, which is the basis for Nginx to support high concurrent connections.

4, with high reliability

Nginx is a highly reliable web server. This is why we chose Nginx. Now many websites use Nginx, which is enough to explain the reliability of Nginx. High reliability comes from the excellent design of its core framework code, the simplicity of the module design, and these modules are very stable.

5, high scalability

Nginx's design is extremely scalable. It consists entirely of modules with different functions, different levels, and different types of coupling. This design created a large number of third-party modules of Nginx.

6, hot deployment

The separation between the master management process and the worker process enables Nginx to have hot-deployment capabilities that can upgrade Nginx executables on a 7x24-hour continuous basis. You can also modify the configuration file, replace the log file, and other functions without stopping the service.

7. Free BSD License Agreement

The BSD license agreement not only allows users to use Nginx for free, it also allows users to modify the Nginx source code, and also allows users to use it for commercial purposes.

How to use Nginx

Nginx installation

Centos 7.

```
wget https://nginx.org/download/nginx-1.13.12.tar.gz

cd nginx-1.13.12.tar.gz

tar -xvf nginx-1.13.12.tar.gz

cd nginx-1.13.12

make & make install

systemctl status nginx
```

After starting Nginx,verify that the installation is successful.

```
Curl http://localhost
```

Nginx.conf is the main configuration file /etc/nginx/nginx.conf. The contents of the default configuration after removing the comment are as follows:

```
worker_process      # Indicates the number of worker processes, typically set to the number of CPU cores

worker_connections  # Indicates the maximum number of connections per worker process

server{}            # The block defines the virtual host

    listen          # Listening port

    server_name     # Listening to the domain name

    location {}     # It is used to configure the matching URI. The URI is "/uri/" in the syntax.

    location /{}    # Match any query because all requests start with /

        root        # Specify the resource search path corresponding to URI, where html is the relative path and the full path is

                    # /opt/nginx-1.7.7/html/

        index       # Specify the name of the home page index file. You can configure more than one, separated by spaces. If there is more

                    # One, in the order of configuration.
```

OpenResty

Setting up the environment

```
$ wget https://openresty.org/download/openresty-1.11.2.2.tar.gz
$ tar -xvf openresty-1.11.2.2.tar.gz
$ cd openresty-1.11.2.2
$ make && make install
```

HelloWorld

HelloWorld is our first ever starter program. But OpenResty is not a programming language. Unlike other programming languages, HelloWorld, let's see what's different.

Create a working directory

After OpenResty is installed, there are configuration files and related directories. For the working directory and installation directory do not interfere with each other, and by the way to learn simple configuration file writing, we also create an OpenResty working directory to practice, and write another configuration file . I chose to create the openresty-test directory in the current user directory and create the logs and conf subdirectories under that directory for the logs and configuration files.

```
$ mkdir openresty-work

$ cd openresty-work/

$ mkdir logs && conf
```

Create a configuration file

In the conf/ directory to create a text file as a configuration file, named nginx.conf, the contents of the file are as follows:

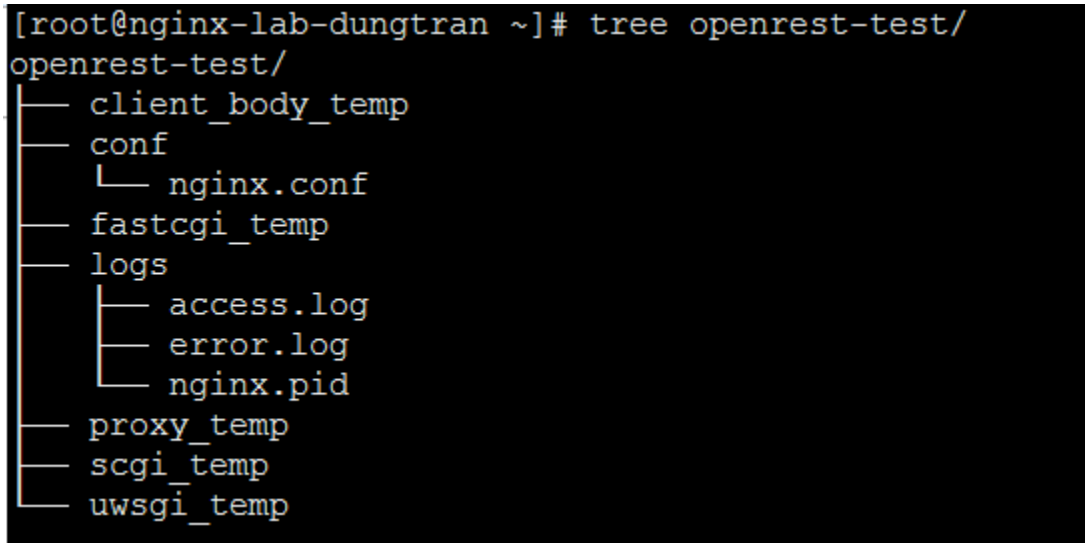
```
worker_processes  1;        # nginx worker number

error_log logs/error.log;   # Specify error log file path

events {

    worker_connections 1024; # Listen port, if your 6699 port is already occupied, you need to modify

}
```

```
http {
    server {

        listen 6699;
        location / {
            default_type text/html;

            content_by_lua_block {
                ngx.say("HelloWorld")
            }
        }
    }
```

All is ready except for the opportunity

After the startup is successful, we can check if the nginx process exists and check the contents of the response by accessing the HTTP page. Assuming we have installed OPENRESY into /usr/local/openresty (default), we make our nginx executable of our OPENRESTY installation available in our PATH environment. The operating tips are as follows:

```
➜ PATH=/usr/local/openresty/nginx/sbin:$PATH
➜ export PATH
➜  ~ nginx -p `pwd`/ -c openresty-test/conf/nginx.conf
➜  ~ ps -ef | grep nginx
  501 88620     1   0 10:58AM ?? 0:00.00 nginx: master process nginx -p
                                    /Users/yuansheng/openresty-test
  501 88622 88620   0 10:58AM ?? 0:00.00 nginx: worker process
➜  ~ curl http://localhost:6699 -i
HTTP/1.1 200 OK
Server: openresty/1.9.7.3
Date: Sun, 20 Mar 2016 03:01:35 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive

HelloWorld
```


[root@nginx-lab-dungtran ~]# tree openrest-test/
openrest-test/
├── client_body_temp
├── conf
│   └── nginx.conf
├── fastcgi_temp
├── logs
│   ├── access.log
│   ├── error.log
│   └── nginx.pid
├── proxy_temp
├── scgi_temp
└── uwsgi_temp