OpenResty分享

实验环境地址

对windows操作系统

ftp://192.168.12.40/pub/experiment.tar.gz

目录

A OpenResty介绍

B 安装配置

C mysql、redis组件

D 共享内存、缓存

E 案例分享--RTB

A OpenResty介绍

OpenResty介绍

特点

OpenResty介绍

Nginx

Nginx 是一个高性能的HTTP和反向代理服务器，也是一个MAP/POP3/SMTP 代理服务器。

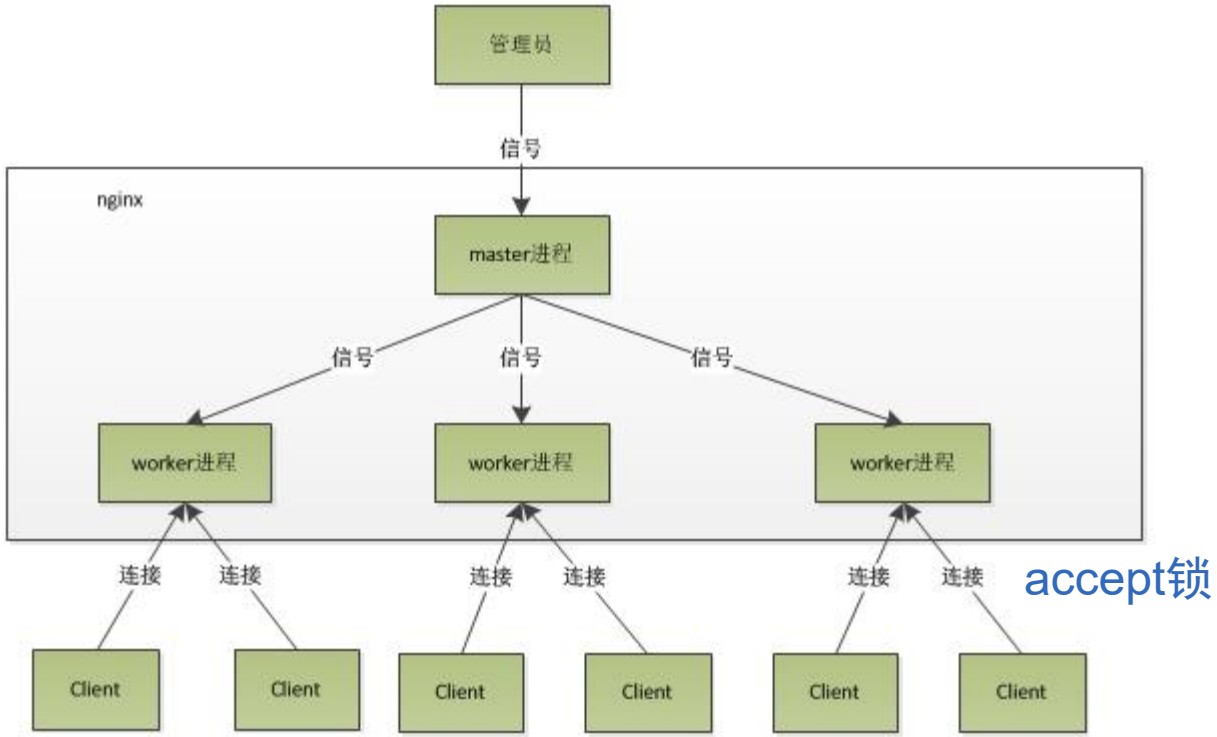
2015年6月，Netcraft 收到的调查网站有 8 亿多家，主流 Web 服务器市场份额（前四名）如下表：

Web服务器	市场占有率
Apache	49.53%
Nginx	13.52%
Microsoft IIS	12.32%
Google Web Server	7.72%

其中在访问量最多的一万个网站中，Nginx 的占有率已超过 Apache。

OpenResty介绍

Nginx进程模型



accept锁

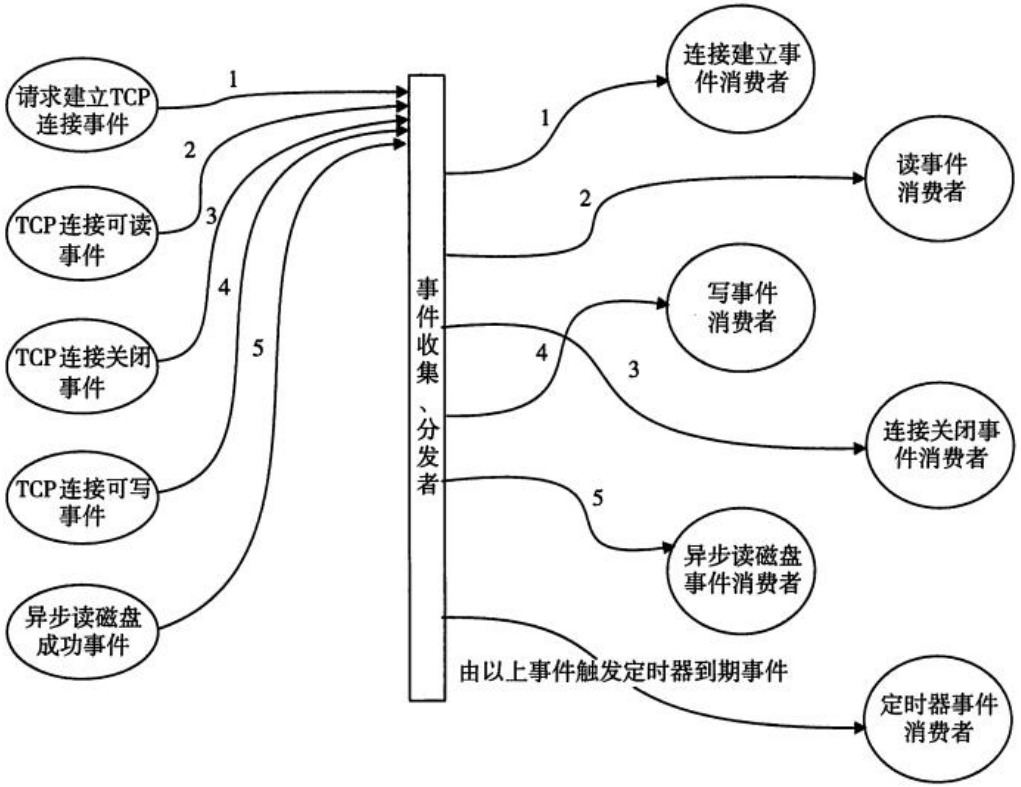
C10K

对于每个请求会独占一个工作线程的web服务器，当并发数上到几千时，就同时有几千的线程在处理请求了。这对操作系统来说，是个不小的挑战，线程带来的内存占用非常大，线程的上下文切换带来的cpu开销很大，自然性能就上不去了，而这些开销完全是没有意义的。

nginx进程数一般等于CPU核数，减少不必要的进程切换

OpenResty介绍

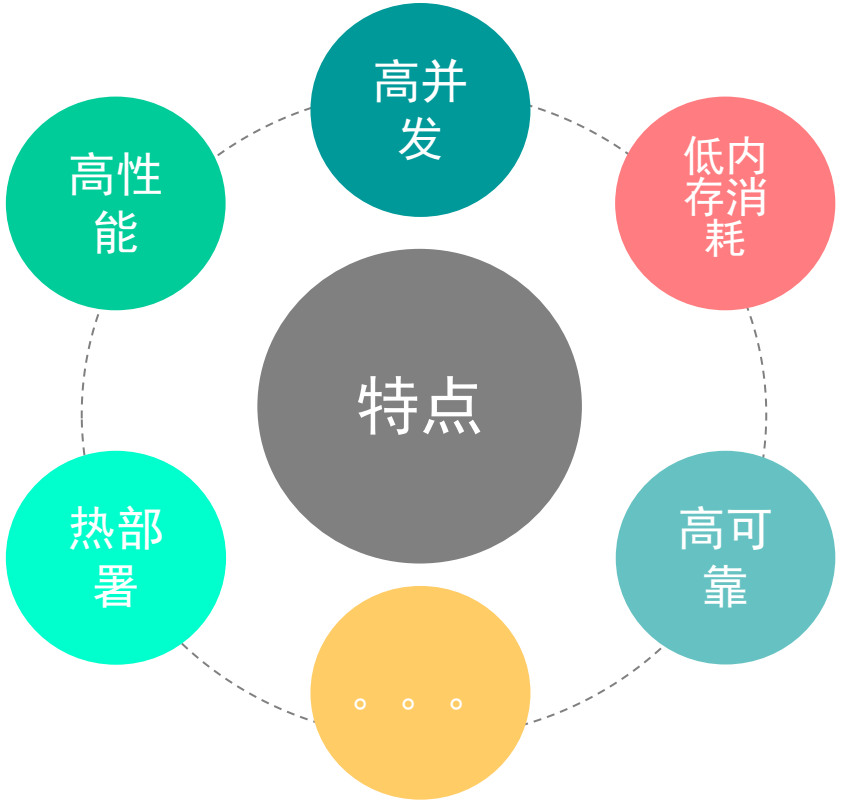
Nginx事件机制



异步非阻塞：epoll

OpenResty介绍

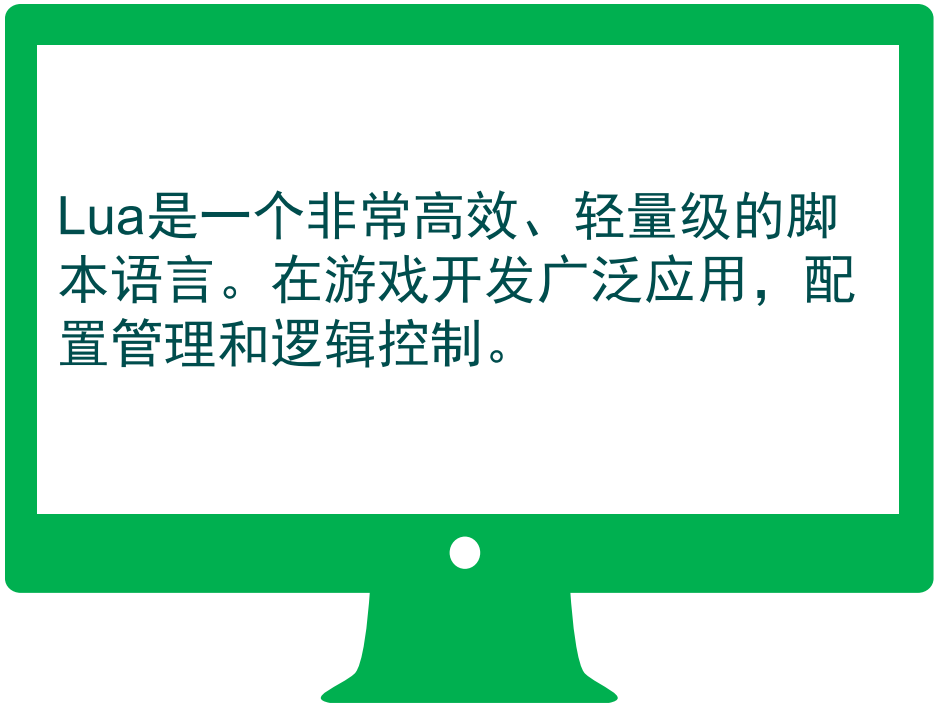
Nginx

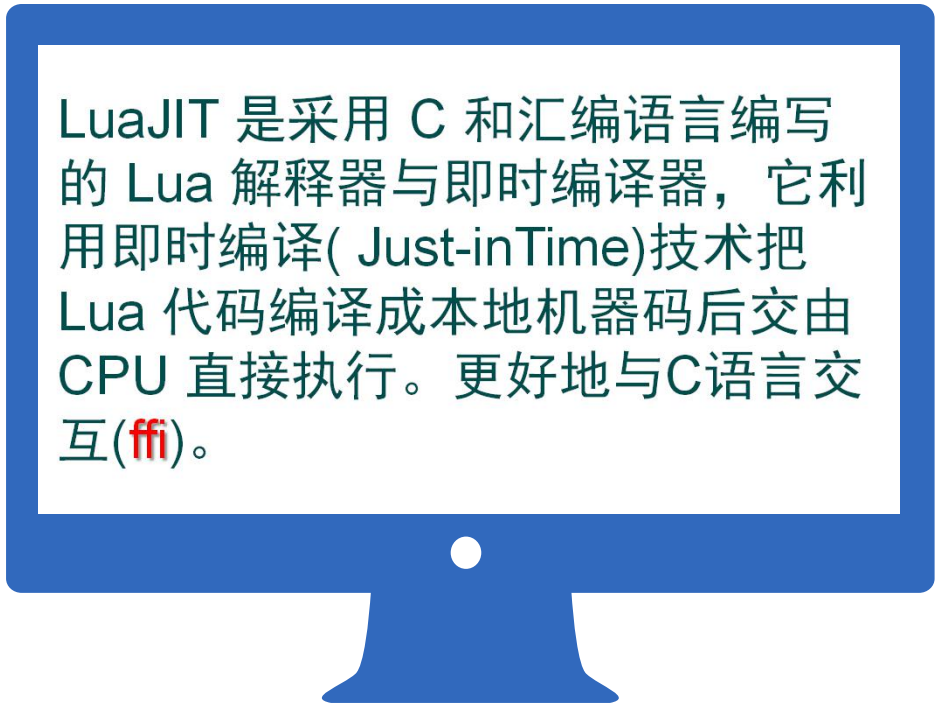


- 反向代理
- web服务器
- 负载均衡
- 静态文件服务
- 访问权限控制
- 限流

OpenResty介绍

Lua vs Luajit





OpenResty介绍

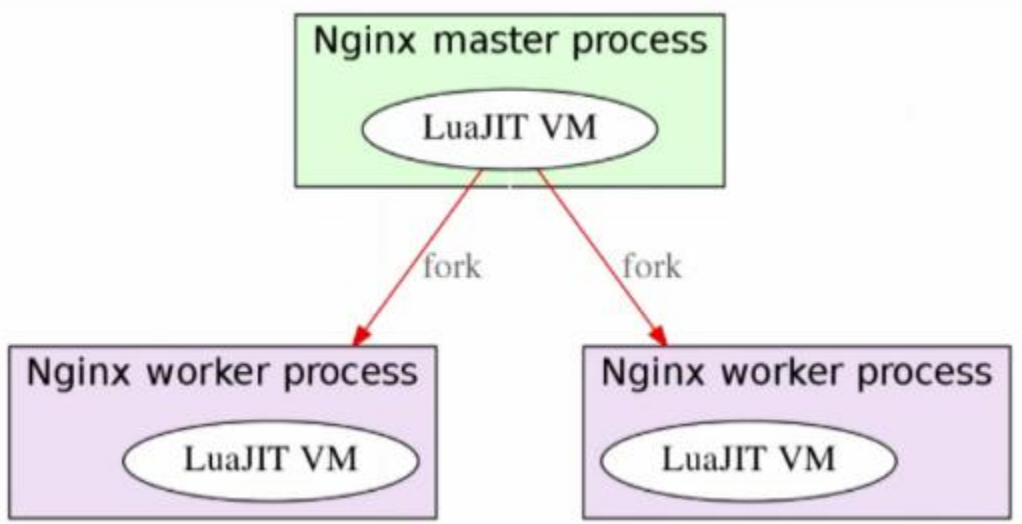
一个Lua程序

```lua
local cjson = require("cjson")

local function print_env()
    print("LUA_PATH", package.path)
    print("LUA_CPATH", package.cpath)
    print()
end

print_env()

local input = [[{"id": "9d66d9249cc5bd549b0e68b9fedc6
                 "storeurl": "https://itunes.apple.co
                 "name": "App Name","bundle": "yourco

local tab = cjson.decode(input)

for k, v in pairs(tab) do
    if type(v) ~= 'table' then
        print(k .. ': ' .. v)
    else
        print(k .. ": ")
        for _, v in ipairs(v) do
            print(v)
        end
    end
    print("----------------------------------------
end

print()

print("tab.id", tab.id)
print("tab['id']", tab['id'])
print()

local input2 = "{'abc': , 'def': 1}"

local status, tab2 = pcall(cjson.decode, input2)
if not status then
    print(status, tab2)
else
    print(status, tab2)
end

print()
local ffi = require("ffi")
ffi.cdef[[
    int printf(const char *fmt, ...);
]]
ffi.C.printf("Hello %s!\n", "world")
```

test.lua [+] [utf-8] 1,1 Top | test.lua [+] [utf-8] 56,1 Bot

OpenResty介绍

OpenResty如何工作



OpenResty介绍

Lua协程

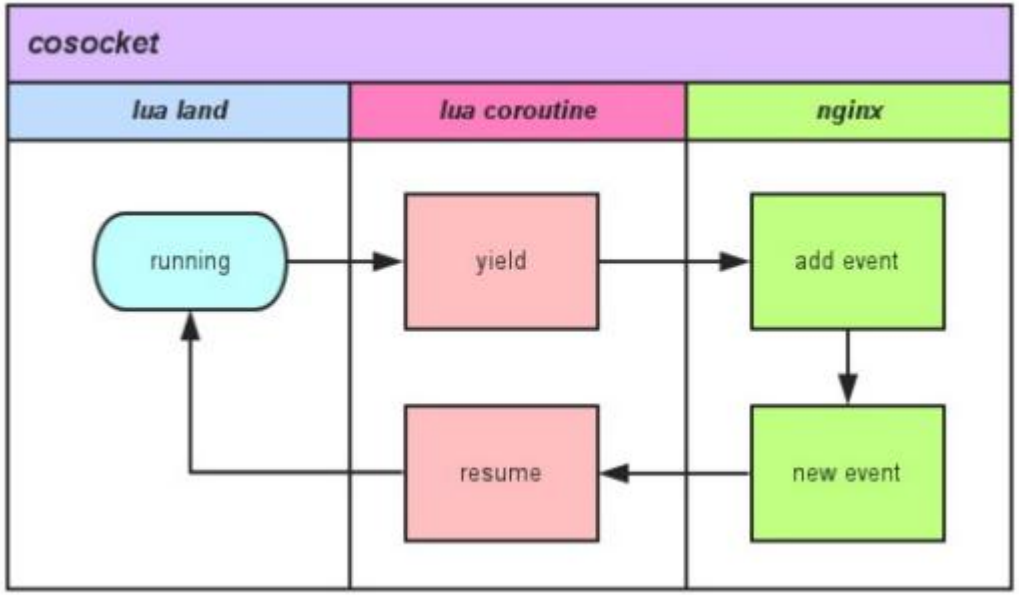
协同程序（coroutine）与多线程情况下的线程比较类似：有自己的堆栈、局部变量、指令指针，但与其它协程共享全局变量等很多信息。

协程类似一种多线程，但与多线程还有很多区别：

1. 协程并非os线程，所以创建、切换开销比线程相对要小。
2. 协程与线程一样有自己的栈、局部变量等，但是协程的栈是在用户进程空间模拟的，所以创建、切换开销很小。
3. 多线程程序是多个线程并发执行，也就是说在一瞬间有多个控制流在执行。而协程强调的是一种多个协程间协作的关系，只有当一个协程主动放弃执行权，另一个协程才能获得执行权，所以在某一瞬间，多个协程间只有一个在运行。
4. 由于多个协程时只有一个在运行，所以对于临界区的访问不需要加锁，而多线程的情况则必须加锁。
5. 多线程程序由于有多个控制流，所以程序的行为不可控，而多个协程的执行是由开发者定义的所以是可控的。

OpenResty介绍

cosocket



1、每个worker（工作进程）创建一个Lua VM，worker内所有协程共享VM；
2、将Nginx I/O原语封装后注入 Lua VM，允许Lua代码直接访问；
3、每个外部请求都由一个Lua协程处理，协程之间数据隔离；
4、Lua代码调用I/O操作等异步接口时，会挂起当前协程（并保护上下文数据），而不阻塞worker；
5、I/O等异步操作完成时还原相关协程上下文数据，并继续运行；

mysql redis memcached等模块都是基于cosocket

OpenResty介绍

Nginx C 模块 vs OpenResty

```
/**********************************************************
 * handler after finish read request body
 **********************************************************/
static ngx_int_t ngx_http_hello_body_handler(ngx_http_request_t * r)
{
    /*read body, return ngx_str_t*/
    /*ngx_str_t body = ngx_http_hello_get_body(r);*/

    ngx_str_t body = ngx_string("hello world\n");

    /*http response*/
    ngx_str_t type = ngx_string( "text/plain" );
    r->headers_out.status = NGX_HTTP_OK;
    r->headers_out.content_length_n = body.len;
    r->headers_out.content_type = type;

    ngx_int_t rc = ngx_http_send_header(r);
    if (rc == NGX_ERROR || rc > NGX_OK || r->header_only) {
        return rc;
    }

    ngx_buf_t * b;
    b = ngx_create_temp_buf(r->pool, body.len);
    if (b == NULL) {
        return NGX_HTTP_INTERNAL_SERVER_ERROR;
    }

    ngx_memcpy(b->pos, body.data, body.len);
    b->last = b->pos + body.len;
    b->last_buf = 1 ;

    ngx_chain_t out;
    out.buf = b;
    out.next = NULL;


    return ngx_http_output_filter(r, & out);
    /*http response end*/
}
```

```
/**********************************************************************
 * get request body
 **********************************************************************/
static ngx_str_t ngx_http_hello_get_body(ngx_http_request_t * r)
{
    u_char * p;
    u_char * data;
    size_t len;
    //unsigned int len;
    ngx_buf_t * buf;
    ngx_buf_t * next;
    ngx_chain_t * cl;
    ngx_str_t body = ngx_null_string;
    if (r->request_body == NULL || r->request_body->bufs == NULL)
    {
        return body;
    }
    /*
       if(r->request_body->temp_file)
       {
       body = r->request_body->temp_file->file.name;
       return body;
       }
       */
    cl = r->request_body->bufs;
    buf = cl->buf;
    if (cl->next == NULL)
    {
        len = buf->last - buf->pos;
        p = ngx_pnalloc(r->pool, len + 1 );
        if (p == NULL)
        {
            return body;
        }
        data = p;
        ngx_memcpy(p, buf->pos, len);
        data[len] = 0 ;
    }
    else
    {
```

```
location /helloworld {
    content_by_lua_block {
        ngx.say("HelloWorld")
    }
}
```

curl http://127.0.0.1/hello

curl http://127.0.0.1/helloworld

OpenResty介绍

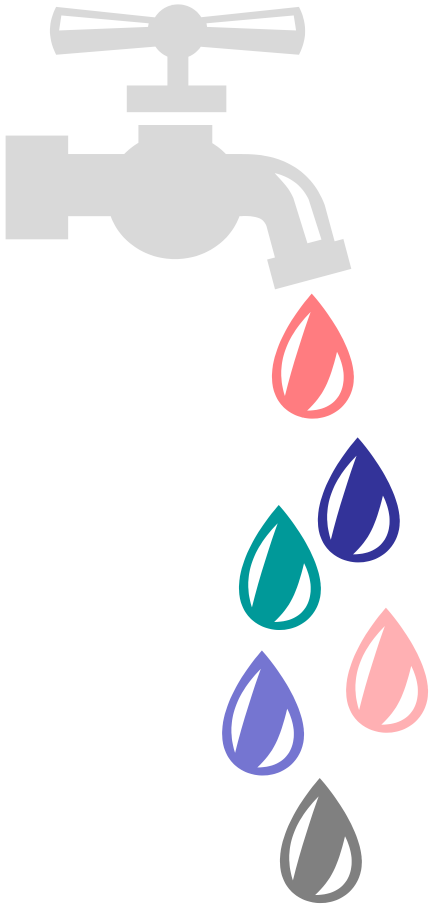
组件

- LuaJIT
- ArrayVarNginxModule
- AuthRequestNginxModule
- CoolkitNginxModule
- DrizzleNginxModule
- EchoNginxModule
- EncryptedSessionNginxModule
- FormInputNginxModule
- HeadersMoreNginxModule
- IconvNginxModule
- StandardLuaInterpreter
- MemcNginxModule
- Nginx
- NginxDevelKit
- LuaCjsonLibrary
- LuaNginxModule
- LuaRdsParserLibrary
- LuaRedisParserLibrary
- LuaRestyCoreLibrary
- LuaRestyDNSLibrary
- LuaRestyLockLibrary
- LuaRestyLrucacheLibrary
- LuaRestyMemcachedLibrary
- LuaRestyMySQLLibrary
- LuaRestyRedisLibrary
- LuaRestyStringLibrary
- LuaRestyUploadLibrary
- LuaRestyUpstreamHealthcheckLibrary
- LuaRestyWebSocketLibrary
- LuaUpstreamNginxModule
- PostgresNginxModule
- RdsCsvNginxModule
- RdsJsonNginxModule
- RedisNginxModule
- Redis2NginxModule
- RestyCLI
- SetMiscNginxModule
- SrcacheNginxModule
- XssNginxModule

B 安装配置

安装配置

相关网址



- https://openresty.org/en/download.html
- https://www.gitbook.com/book/moonbingbing/openresty-best-practices/details
- https://github.com/openresty/lua-nginx-module
- http://nginx.org/en/docs/
- http://luajit.org/index.html
- http://www.lua.org/manual/5.1/

安装配置

安装步骤(ubuntu)

https://openresty.org/en/installation.html

依赖安装

```
sudo apt-get install libreadline-dev libncurses5-dev
libpcre3-dev ibssl-dev perl make build-essential
```

源码解压

```
tar zxvf openresty-1.9.15.1.tar.gz && cd openresty-
1.9.15.1/
```

configure

```
./configure --prefix=/opt/openresty --with-luajit --with-
http_iconv_module -j2
```

编译

```
make -j2
```

安装

```
sudo make install
```

安装配置

命令

- ./configure --help
- nginx -s reload
- nginx -s stop
- nginx -t
- nginx -V

```
curl http://127.0.0.1
```

安装配置

配置

```
user  demo;
worker_processes  auto;
worker_cpu_affinity auto;

error_log  logs/error.log info;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

pid        logs/nginx.pid;

events {
    worker_connections  50000;
    use epoll;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_
    #                  '$status $body_bytes_sent "$http_ref
    #                  '"$http_user_agent" "$http_x_forward

    #access_log  logs/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    lua_package_path '/opt/openresty/nginx/lua_ad/?.lua;/op

    lua_shared_dict cache_ngx 128m;

    lua_shared_dict my_locks 100k;

    lua_shared_dict cache_ad 128m;

    server {
        listen       80;
        server_name  localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            root   html;
            index  index.html index.htm;
        }

        #error_page  404              /404.html;

        # redirect server error pages to the static page /
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }

        location /hello {
            hello;
        }

        location /helloworld {
            content_by_lua_block {
                ngx.say("HelloWorld")
            }
        }

        location /mixed {
            set_by_lua $a 'ngx.log(ngx.INFO, "set_by_lua")
            rewrite_by_lua 'ngx.log(ngx.INFO, "rewrite_by_
            access_by_lua 'ngx.log(ngx.INFO, "access_by_lu
            header_filter_by_lua 'ngx.log(ngx.INFO, "heade
            body_filter_by_lua 'ngx.log(ngx.INFO, "body_fi
            log_by_lua 'ngx.log(ngx.INFO, "log_by_lua")';
            content_by_lua 'ngx.log(ngx.INFO, "content_by_
        }

        location /ngx_ctx {
            rewrite_by_lua '
                ngx.ctx.foo = 76
            ';
            access_by_lua '
                ngx.ctx.foo = ngx.ctx.foo + 3
            ';
            content_by_lua_block {
                ngx.say(ngx.ctx.foo)
            }
        }

        location /mysql_test {
            content_by_lua_file lua/mysql_test.lua;
        }

        location /mysql_select {
            content_by_lua_file lua/mysql_select.lua;
        }

        location /redis_set {
            content_by_lua_file lua/redis_set.lua;
        }

        location /redis_get {
            content_by_lua_file lua/redis_get.lua;
        }

        location /iredis_get {
            content_by_lua_file lua/iredis_get.lua;
        }

        location /iredis_get2 {
            content_by_lua_file lua/iredis_get2.lua;
        }

        location /iredis_get3 {
            content_by_lua_file lua/iredis_get3.lua;
        }

        location ~ ^/api/([-_a-zA-Z0-9/]+) {
            access_by_lua_file lua/access_check.lua;
            content_by_lua_file lua/$1.lua;
        }

        location /adbid {
            content_by_lua_file lua_ad/adbid.lua;
        }

    }
}
```

C mysql、redis组件

mysql、redis组件

执行阶段

01 set_by_lua
流程分支处理判断、变量初始化

02 rewrite_by_lua
转发、重定向

03 access_by_lua
IP准入、接口权限

04 content_by_lua
内容生成

05 header_filter_by_lua
应答http过滤处理

06 body_filter_by_lua
应答body过滤处理

07 log_by_lua
会话完成后本地异步完成日志记录

mysql、redis组件

阶段之间传递变量

ngx.ctx

```
location /mixed {
    set_by_lua $a 'ngx.log(ngx.INFO, "set_by_lua")';
    rewrite_by_lua 'ngx.log(ngx.INFO, "rewrite_by_lua")';
    access_by_lua 'ngx.log(ngx.INFO, "access_by_lua")';
    header_filter_by_lua 'ngx.log(ngx.INFO, "header_filter_by_lua")';
    body_filter_by_lua 'ngx.log(ngx.INFO, "body_filter_by_lua")';
    log_by_lua 'ngx.log(ngx.INFO, "log_by_lua")';
    content_by_lua 'ngx.log(ngx.INFO, "content_by_lua")';
}
```

```
location /ngx_ctx {
    rewrite_by_lua '
        ngx.ctx.foo = 76
    ';
    access_by_lua '
        ngx.ctx.foo = ngx.ctx.foo + 3
    ';
    content_by_lua_block {
        ngx.say(ngx.ctx.foo)
    }
}
```

curl http://127.0.0.1/mixed

curl http://127.0.0.1/ngx_ctx

mysql、redis组件

mysql--建表

```lua
local mysql = require "resty.mysql"
local db, err = mysql:new()
if not db then
    ngx.say("failed to instantiate mysql: ", err)
    return
end

db:set_timeout(1000) -- 1 sec

local ok, err, errno, sqlstate = db:connect{
    host = "127.0.0.1",
    port = 3306,
    database = "ngx_test",
    user = "root",
    password = "111111",
    max_packet_size = 1024 * 1024
}

if not ok then
    ngx.say("failed to connect: ", err, ": ", errno, " ", sql
    return
end

local res, err, errno, sqlstate =
    db:query("drop table if exists user")
if not res then
    ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate,
    return
end

local res, err, errno, sqlstate =
    db:query("create table user "
             .. "(id serial primary key, "
             .. "name varchar(8), "
             .. "email varchar(32), "
             .. "password varchar(64), "
             .. "index idx_name (name))")
if not res then
    ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate,
    return
end

ngx.say("table user created.")

local res, err, errno, sqlstate =
    db:query("insert into user (name, email, password) "
             .. "values (\'zhuyu\', \'zhuyu1989.hi@gmail.com\
if not res then
    ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate,
    return
end

ngx.say(res.affected_rows, " rows inserted into table cats ",
        "(last insert id: ", res.insert_id, ")")

-- run a select query
-- the result set:
local res, err, errno, sqlstate =
    db:query("select * from user")
if not res then
    ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate,
    return
end

local cjson = require "cjson"
ngx.say("result: ", cjson.encode(res))

-- put it into the connection pool of size 100,
-- with 10 seconds max idle timeout
local ok, err = db:set_keepalive(10000, 100)
if not ok then
    ngx.say("failed to set keepalive: ", err)
    return
end
```

mysql create.lua [utf-8] 1,1 Top　mysql create.lua [utf-8] 74,1 Bot

curl http://127.0.0.1/mysql_create

mysql、redis组件

mysql--insert

```
local arg = ngx.req.get_uri_args()


local res, err, errno, sqlstate =
    db:query(string.format([[insert into user (name, email, password)
                             values ('%s', '%s', '%s')]], arg.name, arg.email, arg.password))
if not res then
    ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate, ".")
    return
end

ngx.say(res.affected_rows, " rows inserted into table cats ",
        "(last insert id: ", res.insert_id, ")")
```

curl 'http://127.0.0.1/mysql_insert?name=jack&email=jack@163.com&password=iefioweio'

mysql、redis组件

mysql--select

```
local arg = ngx.req.get_uri_args()

-- run a select query
-- the result set:
local res, err, errno, sqlstate =
    db:query(string.format("select * from user where name='%s'", arg.name))
if not res then
    ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate, ".")
    return
end

local cjson = require "cjson"
ngx.say("result: ", cjson.encode(res))
```

curl http://127.0.0.1/mysql_select?name=zhuyu

ab -k -c 100 -n 100000 http://127.0.0.1/mysql_select?name=zhuyu

mysql、redis组件

redis

```lua
local redis = require "resty.redis"
local red = redis:new()

red:set_timeout(1000) -- 1 sec

local ok, err = red:connect("127.0.0.1", 6379)
if not ok then
    ngx.say("failed to connect: ", err)
    return
end

local arg = ngx.req.get_uri_args()

local ok, err = red:set(arg.key, arg.value)
if not ok then
    ngx.say("failed to set: ", err)
    return
end

ngx.say("set result: ", ok)

-- put it into the connection pool of size 100,
-- with 10 seconds max idle time

local ok, err = red:set_keepalive(10000, 100)
if not ok then
    ngx.say("failed to set keepalive: ", err)
    return
end
```

redis_set.lua

```lua
local redis = require "resty.redis"
local red = redis:new()

red:set_timeout(1000) -- 1 sec

local ok, err = red:connect("127.0.0.1", 6379)
if not ok then
    ngx.say("failed to connect: ", err)
    return
end

local arg = ngx.req.get_uri_args()

local value, err = red:get(arg.key)
if not value then
    ngx.say("failed to get: ", err)
    return
end

ngx.say(arg.key , ': ', value)

-- put it into the connection pool of size 100,
-- with 10 seconds max idle time

local ok, err = red:set_keepalive(10000, 100)
if not ok then
    ngx.say("failed to set keepalive: ", err)
    return
end
```

redis_get.lua

```
curl 'http://127.0.0.1/redis_set?key=ad0001&value=插屏广告'
curl 'http://127.0.0.1/redis_get?key=ad0001'

ab -n 500000 -c 100 -k 'http://127.0.0.1/redis_set?key=ad0001&value=插屏广告'
ab -n 500000 -c 100 -k 'http://127.0.0.1/redis_get?key=ad0001'
```

D 共享内存、缓存

共享内存、缓存

缓存

进程间共享内存 shared_dict

```
lua_shared_dict cache_ngx 128m;
```

ab -n 500000 -c 100 -k
http://127.0.0.1/iredis_get

ab -n 500000 -c 100 -k
http://127.0.0.1/iredis_get2

```lua
local redis = require "resty.iredis"

local function get_from_redis(key)
    local red = redis:new({["ip"]="127.0.0.1", ["port"]=6379})
    local res, err = red:get(key)
    return res
end

local function set_to_cache(key, value, exptime)
    if not exptime then
        exptime = 0
    end
    local cache_ngx = ngx.shared.cache_ngx
    local succ, err, farcible = cache_ngx:set(key, value, exptime)
    return succ
end

local function get_from_cache(key)
    local cache_ngx = ngx.shared.cache_ngx
    local value = cache_ngx:get(key)
    if not value then
        value = get_from_redis(key)
        set_to_cache(key, value, 100)
    end
    return value
end

local res = get_from_cache('ad0001')
ngx.say(res)
```

iredis get2.lua

进程内缓存 lrucache

共享内存、缓存

缓存

缓存失效风暴

加锁lua-resty-lock

```
lua_shared_dict my_locks 100k;
```

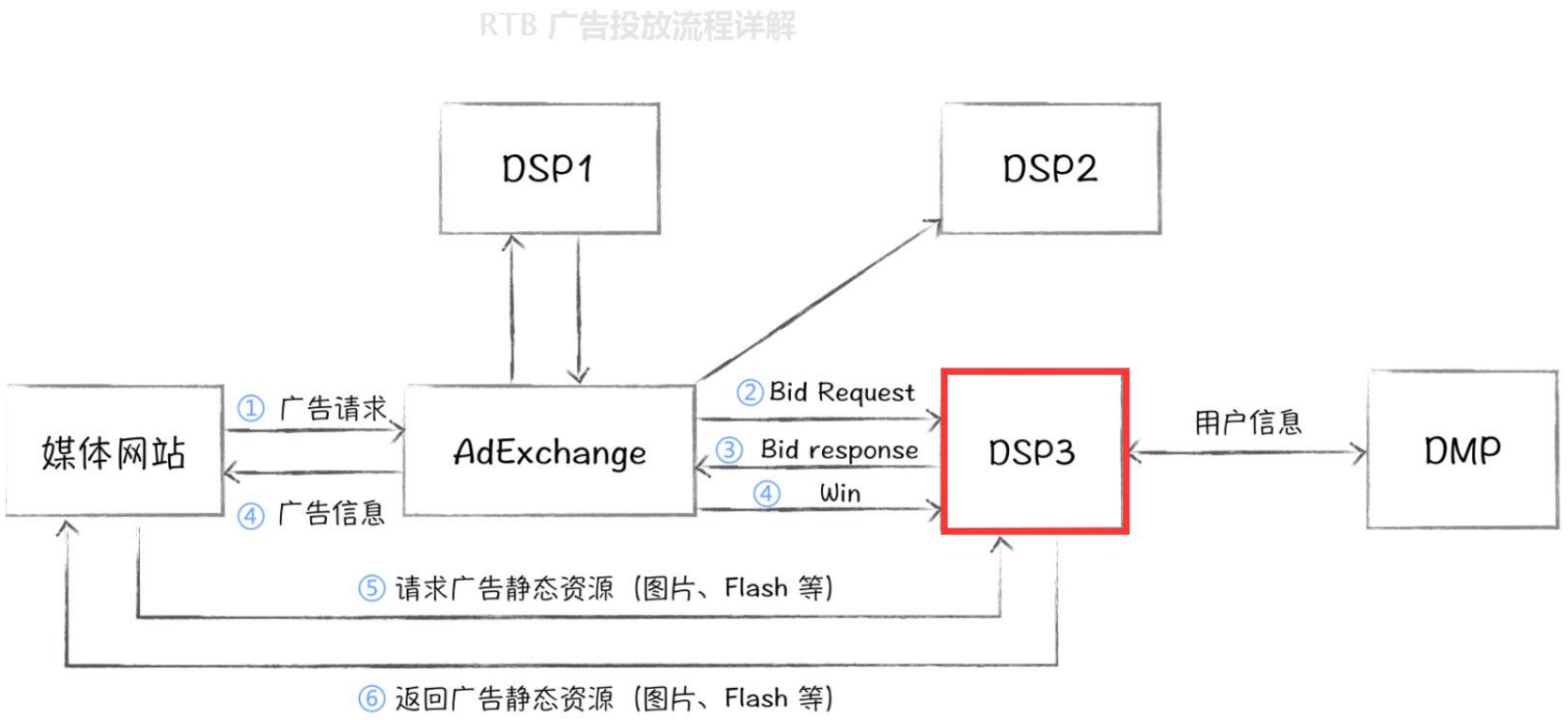
```lua
local redis = require "resty.iredis"

local function get_from_redis(key)
    local red = redis:new({["ip"]="127.0.0.1", ["port"]=6379})
    local res, err = red:get(key)
    return res
end

local function set_to_cache(key, value, exptime)
    if not exptime then
        exptime = 0
    end
    local cache_ngx = ngx.shared.cache_ngx
    local succ, err, farcible = cache_ngx:set(key, value, exptime)
    return succ
end

local function get_from_cache(key)
    local cache_ngx = ngx.shared.cache_ngx
    local value = cache_ngx:get(key)
    if not value then
        local lock = require "resty.lock"
        local lock = lock:new("my_locks")
        lock:lock("my_key")
        value = get_from_redis(key)
        lock:unlock()

        set_to_cache(key, value, 100)
    end
    return value
end

local res = get_from_cache('ad0001')
ngx.say(res)

```

iredis_get3.lua

E 案例分享--RTB

案例分享--RTB

RTB


RTB 广告投放流程详解
DSP1
DSP2
媒体网站
① 广告请求
④ 广告信息
AdExchange
② Bid Request
③ Bid response
④ Win
DSP3
用户信息
DMP
⑤ 请求广告静态资源（图片、Flash 等）
⑥ 返回广告静态资源（图片、Flash 等）

案例分享--RTB

DSP处理流程

01 读取广告竞价请求报文体，并用cjson将报文转成table

02 读取shared dict的缓存，如果缓存为空或者缓存失效，重新缓存mysql相关表

03 根据请求报文，筛选缓存中符合条件的广告

04 从符合条件的广告中选择一个广告出价

05 组返回报文，用cjson转成字符串返回

案例分享--RTB

Python vs OpenResty

mysql：存放广告信息、广告投放计划、广告预算

redis： ip-城市、标签库、实时统计已用金额

Python

mysql数据缓存到本地内存

AWS 8核 16G

3000 QPS

OpenResty

mysql数据缓存到shared_dict

本地虚拟机 4核 4G

12000+ QPS

End