

OpenResty分享



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

Nginx + Lua

1

大量精良的Lua库、
第三方模块

2

3

异步非阻塞

4

同步的代码逻辑写
异步

OpenResty介绍



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：

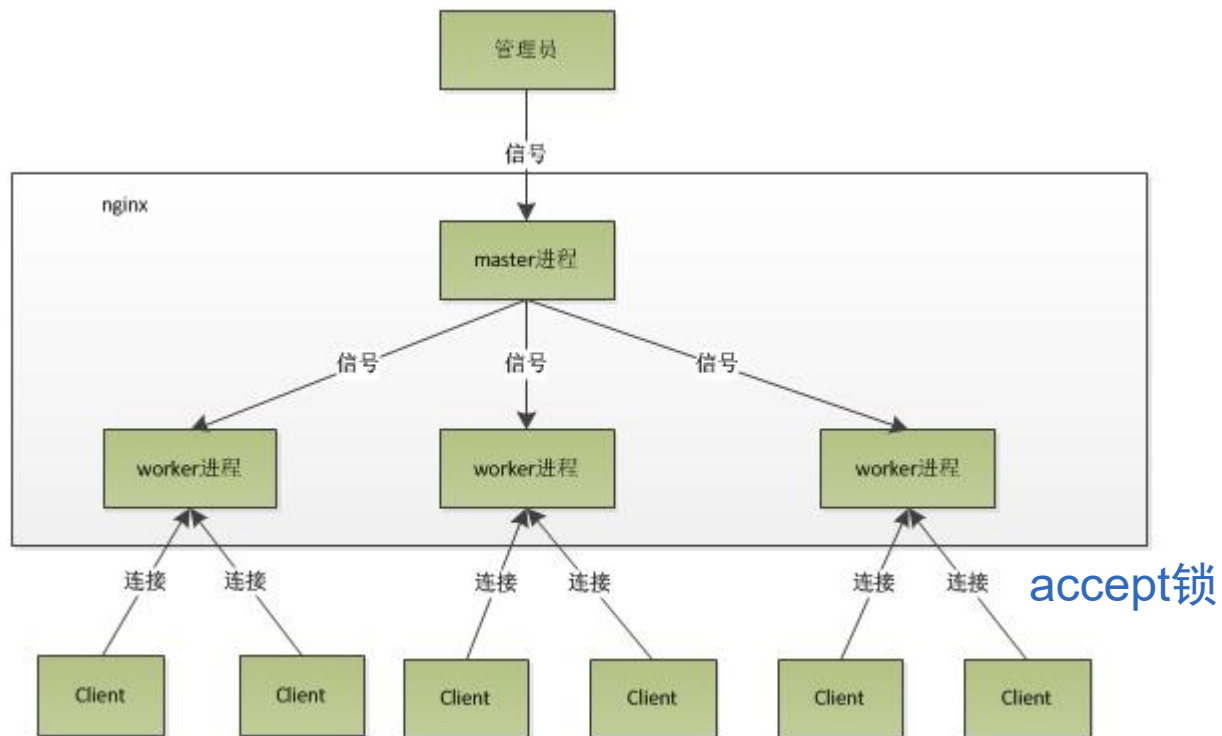
Web服务器	市场占有率
Apache	49.53%
Nginx	13.52%
Microsoft IIS	12.32%
Google Web Server	7.72%

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



C10K

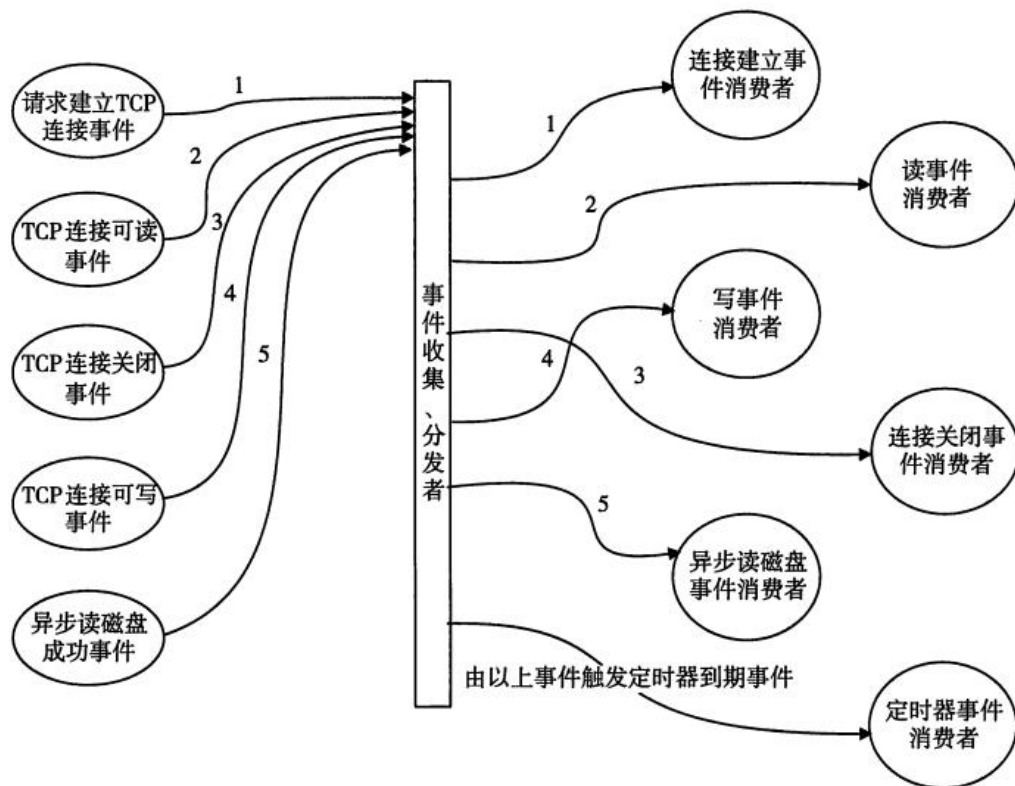
对于每个请求会独占一个工作线程的web服务器，当并发数上到几千时，就同时有几千的线程在处理请求了。这对操作系统来说，是个不小的挑战，线程带来的内存占用非常大，线程的上下文切换带来的cpu开销很大，自然性能就上不去了，而这些开销完全是没有意义的。

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

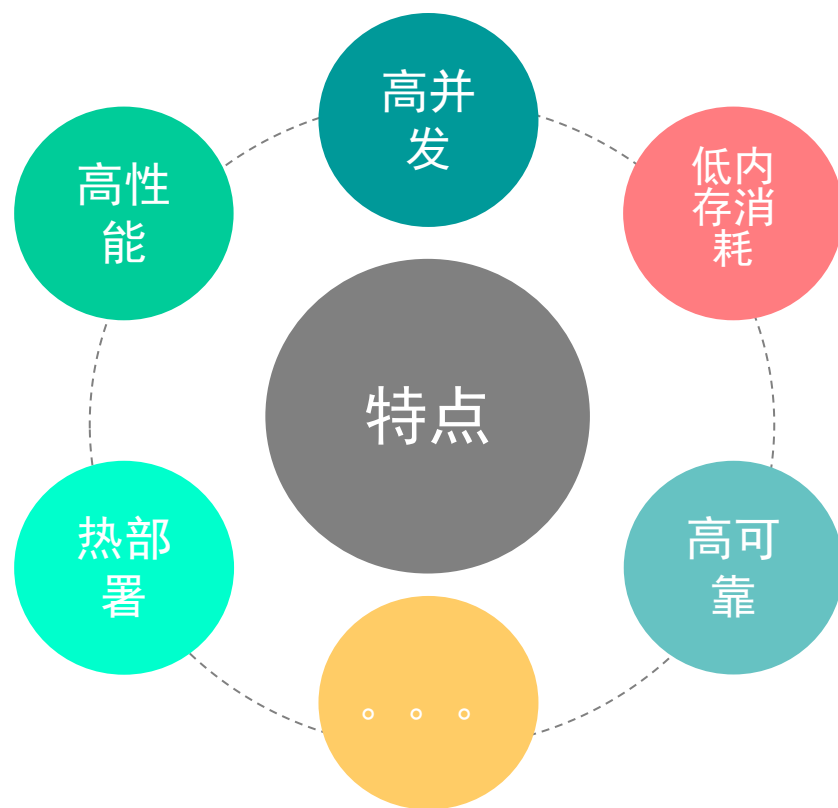


异步非阻塞：epoll 事件循环

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

Lua是一个非常高效、轻量级的脚本语言。在游戏开发广泛应用，配置管理和逻辑控制。

LuaJIT 是采用 C 和汇编语言编写的 Lua 解释器与即时编译器，它利用即时编译(Just-inTime)技术把 Lua 代码编译成本地机器码后交由 CPU 直接执行。更好地与C语言交互(**ffi**)。

OpenResty介绍



一个Lua程序

```
1 local cjson = require("cjson")
2
3 local function print_env()
4     print("LUA_PATH", package.path)
5     print("LUA_CPATH", package.cpath)
6     print()
7 end
8
9 print_env()
10
11 local input = [{"id": "9d66d9249cc5bd549b0e68b9fedc6",
12                 "storeurl": "https://itunes.apple.com",
13                 "name": "App Name", "bundle": "yourco"}]
14
15 local tab = cjson.decode(input)
16
17 for k, v in pairs(tab) do
18     if type(v) ~= 'table' then
19         print(k .. ': ' .. v)
20     else
21         print(k .. ": ")
22         for _, v in ipairs(v) do
23             print(v)
24         end
25     end
26     print("-----")
27 end
28
```

test.lua [+]

[utf-8] 1,1

Top

```
29 print()
30
31 print("tab.id", tab.id)
32 print("tab['id']", tab['id'])
33 print()
34
35 local input2 = '{"abc': , 'def': 1}"
36
37 local status, tab2 = pcall(cjson.decode, input2)
38 if not status then
39     print(status, tab2)
40 else
41     print(status, tab2)
42 end
43
44 print()
45 local ffi = require("ffi")
46 ffi.cdef[[
47     int printf(const char *fmt, ...);
48 ]]
49 ffi.C.printf("Hello %s!\n", "world")
50
51
52
53
54
55
56
```

test.lua [+]

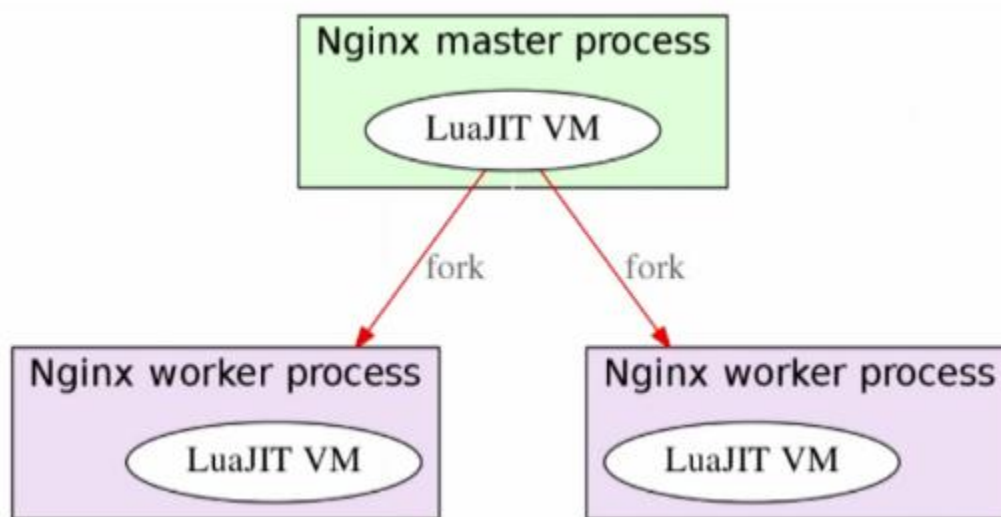
[utf-8] 56,1

Bot

OpenResty介绍



OpenResty如何工作





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。

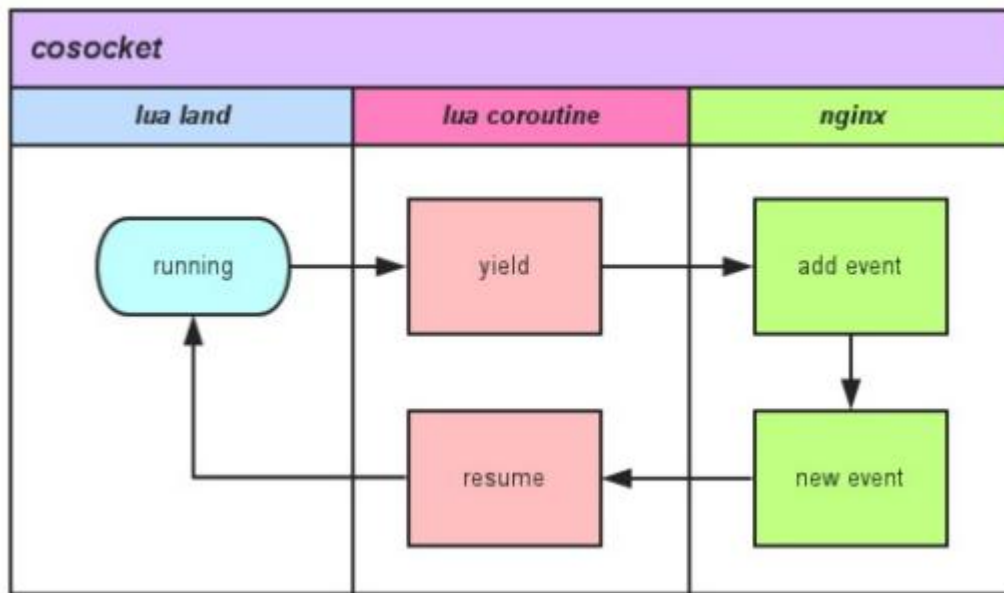
协程类似一种多线程，但与多线程还有很多区别：

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

```
34 /*****  
33  * handler after finish read request body  
32  *****/  
31 static ngx_int_t ngx_http_hello_body_handler(ngx_http_request_t * r)  
30 {  
29     /*read body, return ngx_str_t*/  
28     /*ngx_str_t body = ngx_http_hello_get_body(r);*/  
27  
26     ngx_str_t body = ngx_string("hello world\n");  
25  
24     /*http response*/  
23     ngx_str_t type = ngx_string("text/plain");  
22     r->headers_out.status = NGX_HTTP_OK;  
21     r->headers_out.content_length_n = body.len;  
20     r->headers_out.content_type = type;  
19  
18     ngx_int_t rc = ngx_http_send_header(r);  
17     if (rc == NGX_ERROR || rc > NGX_OK || r->header_only) {  
16         return rc;  
15     }  
14  
13     ngx_buf_t * b;  
12     b = ngx_create_temp_buf(r->pool, body.len);  
11     if (b == NULL) {  
10         return NGX_HTTP_INTERNAL_SERVER_ERROR;  
9     }  
8  
7     ngx_memcpy(b->pos, body.data, body.len);  
6     b->last = b->pos + body.len;  
5     b->last_buf = 1;  
4  
3     ngx_chain_t out;  
2     out.buf = b;  
1     out.next = NULL;  
164  
1  
2     return ngx_http_output_filter(r, & out);  
3     /*http response end*/  
4 }  
5
```

```
34 /*****  
33  * get request body  
32  *****/  
31 static ngx_str_t ngx_http_hello_get_body(ngx_http_request_t * r)  
30 {  
29     u_char * p;  
28     u_char * data;  
27     size_t len;  
26     /*unsigned int len;  
25     ngx_buf_t * buf;  
24     ngx_buf_t * next;  
23     ngx_chain_t * cl;  
22     ngx_str_t body = ngx_null_string;  
21     if (r->request_body == NULL || r->request_body->bufs == NULL)  
20     {  
19         return body;  
18     }  
17     /*  
16     if(r->request_body->temp_file)  
15     {  
14         body = r->request_body->temp_file->file.name;  
13         return body;  
12     }  
11     */  
10     cl = r->request_body->bufs;  
9     buf = cl->buf;  
8     if (cl->next == NULL)  
7     {  
6         len = buf->last - buf->pos;  
5         p = ngx_pnalloc(r->pool, len + 1);  
4         if (p == NULL)  
3         {  
2             return body;  
1         }  
204  
1         data = p;  
2         ngx_memcpy(p, buf->pos, len);  
3         data[len] = 0;  
4     }  
5     else
```

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

curl http://127.0.0.1/hello

curl
http://127.0.0.1/hel
loworld

OpenResty介绍



组件

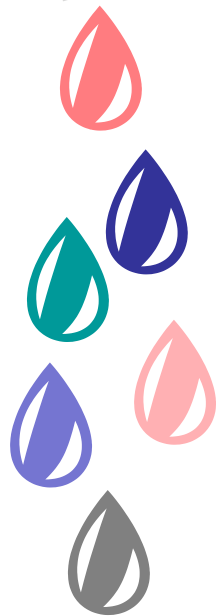
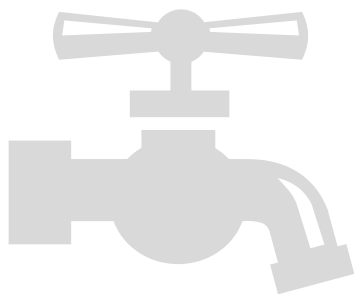
- [LuaJIT](#)
- [ArrayVarNginxModule](#)
- [AuthRequestNginxModule](#)
- [CoolkitNginxModule](#)
- [DrizzleNginxModule](#)
- [EchoNginxModule](#)
- [EncryptedSessionNginxModule](#)
- [FormInputNginxModule](#)
- [HeadersMoreNginxModule](#)
- [IconvNginxModule](#)
- [StandardLuaInterpreter](#)
- [MemcNginxModule](#)
- [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 libssl-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

```
1 user demo;
2 worker_processes auto;
3 worker_cpu_affinity auto;
4 error_log logs/error.log info;
5 #error_log logs/error.log notice;
6 #error_log logs/error.log info;
7
8 pid logs/nginx.pid;
9
10 events {
11     worker_connections 50000;
12     use epoll;
13 }
14
15 http {
16     include mime.types;
17     default_type application/octet-stream;
18
19     #log_format main '$remote_addr - $remote_user [$time_
20     # '$status $body_bytes_sent "$http_ref
21     # '$http_user_agent' "$http_x_forward
22
23     #access_log logs/access.log main;
24
25     sendfile on;
26     #tcp_nopush on;
27
28     #keepalive_timeout 0;
29     keepalive_timeout 65;
30
31     #gzip on;
32
33     lua_package_path '/opt/openresty/nginx/lua_ad/*.lua;/op
34
35     lua_shared_dict cache ngx 128m;
36
37     lua_shared_dict my_locks 100k;
38
39     lua_shared_dict cache_ad 128m;
40
41     server {
42         listen 80;
43         server_name localhost;
44
45         #charset koi8-r;
46
47         #access_log logs/host.access.log main;
48
49         location / {
50             root html;
51             index index.html index.htm;
52         }
53
54         #error_page 404 /404.html;
55
56         # redirect server error pages to the static page /
57         #
58         error_page 500 502 503 504 /50x.html;
59         location = /50x.html {
60             root html;
61         }
62
63         location /hello {
64             hello;
65         }
66
67         location /helloworld {
68             content_by_lua_block {
69                 ngx.say("HelloWorld")
70             }
71         }
72
73         location /mixed {
74             set_by_lua $a 'ngx.log(ngx.INFO, "set_by_lua");
75             rewrite_by_lua 'ngx.log(ngx.INFO, "rewrite_by_
76             access_by_lua 'ngx.log(ngx.INFO, "access_by_lua
77             header_filter_by_lua 'ngx.log(ngx.INFO, "heade
78             body_filter_by_lua 'ngx.log(ngx.INFO, "body_fi
79             log_by_lua 'ngx.log(ngx.INFO, "log_by_lua");';
80             content_by_lua 'ngx.log(ngx.INFO, "content_by_
81         }
82
83         location /ngx_ctx {
84             rewrite_by_lua '
85                 ngx.ctx.foo = 76
86             ';
87             access_by_lua '
88                 ngx.ctx.foo = ngx.ctx.foo + 3
89             ';
90             content_by_lua_block {
91                 ngx.say(ngx.ctx.foo)
92             }
93         }
94
95         location /mysql_test {
96             content_by_lua_file lua/mysql_test.lua;
97         }
98
99         location /mysql_select {
100             content_by_lua_file lua/mysql_select.lua;
101         }
102
103         location /redis_set {
104             content_by_lua_file lua/redis_set.lua;
105         }
106
107         location /redis_get {
108             content_by_lua_file lua/redis_get.lua;
109         }
110
111         location /iredis_get {
112             content_by_lua_file lua/iredis_get.lua;
113         }
114
115         location /iredis_get2 {
116             content_by_lua_file lua/iredis_get2.lua;
117         }
118
119         location /iredis_get3 {
120             content_by_lua_file lua/iredis_get3.lua;
121         }
122
123         location ~ ^/api/([-_a-zA-Z0-9/]+) {
124             access_by_lua_file lua/access_check.lua;
125             content_by_lua_file lua/$1.lua;
126         }
127
128         location /adbid {
129             content_by_lua_file lua_ad/adbid.lua;
130         }
131     }
132 }
133
134 }
```

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/nginx_ctx

mysql、redis组件



mysql--建表

```
1 local mysql = require "resty.mysql"
2 local db, err = mysql:new()
3 if not db then
4     ngx.say("failed to instantiate mysql: ", err)
5     return
6 end
7
8 db:set_timeout(1000) -- 1 sec
9
10 local ok, err, errno, sqlstate = db:connect{
11     host = "127.0.0.1",
12     port = 3306,
13     database = "ngx_test",
14     user = "root",
15     password = "111111",
16     max_packet_size = 1024 * 1024
17 }
18
19 if not ok then
20     ngx.say("failed to connect: ", err, ": ", errno, " ", sqlstate)
21     return
22 end
23
24 local res, err, errno, sqlstate =
25     db:query("drop table if exists user")
26 if not res then
27     ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate)
28     return
29 end
30
31 local res, err, errno, sqlstate =
32     db:query("create table user "
33         .. "(id serial primary key, "
34         .. "name varchar(8), "
35         .. "email varchar(32), "
36         .. "password varchar(64), "
37         .. "index idx_name (name))")
38 if not res then
39     ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate)
40     return
41 end
42
43 ngx.say("table user created.")
44
45 local res, err, errno, sqlstate =
46     db:query("insert into user (name, email, password) "
47         .. "values ('zhuyu', 'zhuyu1989.hi@gmail.com', "
48 if not res then
49     ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate)
50     return
51 end
52
53 ngx.say(res.affected_rows, " rows inserted into table cats ",
54     "(last insert id: ", res.insert_id, ")")
55
56 -- run a select query
57 -- the result set:
58 local res, err, errno, sqlstate =
59     db:query("select * from user")
60 if not res then
61     ngx.say("bad result: ", err, ": ", errno, ": ", sqlstate)
62     return
63 end
64
65 local cJSON = require "cjson"
66 ngx.say("result: ", cJSON.encode(res))
67
68 -- put it into the connection pool of size 100,
69 -- with 10 seconds max idle timeout
70 local ok, err = db:set_keepalive(10000, 100)
71 if not ok then
72     ngx.say("failed to set keepalive: ", err)
73     return
74 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

```
1 local redis = require "resty.redis"
2 local red = redis:new()
3
4 red:set_timeout(1000) -- 1 sec
5
6 local ok, err = red:connect("127.0.0.1", 6379)
7 if not ok then
8     ngx.say("failed to connect: ", err)
9     return
10 end
11
12 local arg = ngx.req.get_uri_args()
13
14 local ok, err = red:set(arg.key, arg.value)
15 if not ok then
16     ngx.say("failed to set: ", err)
17     return
18 end
19
20 ngx.say("set result: ", ok)
21
22 -- put it into the connection pool of size 100,
23 -- with 10 seconds max idle time
24
25 local ok, err = red:set_keepalive(10000, 100)
26 if not ok then
27     ngx.say("failed to set keepalive: ", err)
28     return
29 end
```

redis_set.lua

```
1 local redis = require "resty.redis"
2 local red = redis:new()
3
4 red:set_timeout(1000) -- 1 sec
5
6 local ok, err = red:connect("127.0.0.1", 6379)
7 if not ok then
8     ngx.say("failed to connect: ", err)
9     return
10 end
11
12 local arg = ngx.req.get_uri_args()
13
14 local value, err = red:get(arg.key)
15 if not value then
16     ngx.say("failed to get: ", err)
17     return
18 end
19
20 ngx.say(arg.key , ': ', value)
21
22 -- put it into the connection pool of size 100,
23 -- with 10 seconds max idle time
24
25 local ok, err = red:set_keepalive(10000, 100)
26 if not ok then
27     ngx.say("failed to set keepalive: ", err)
28     return
29 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
```

进程内缓存 lru_cache

```
1 local redis = require "resty.iredis"  
2  
3 local function get_from_redis(key)  
4     local red = redis:new({"ip"]="127.0.0.1", ["port"]=6379})  
5     local res, err = red:get(key)  
6     return res  
7 end  
8  
9 local function set_to_cache(key, value, exptime)  
10     if not exptime then  
11         exptime = 0  
12     end  
13     local cache ngx = ngx.shared.cache_ngx  
14     local succ, err, forcible = cache_ngx:set(key, value, exptime)  
15     return succ  
16 end  
17  
18 local function get_from_cache(key)  
19     local cache ngx = ngx.shared.cache_ngx  
20     local value = cache_ngx:get(key)  
21     if not value then  
22         value = get_from_redis(key)  
23         set_to_cache(key, value, 100)  
24     end  
25     return value  
26 end  
27  
28 local res = get_from_cache('ad0001')  
29 ngx.say(res)  
iredis get2.lua
```

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;
```

```
1 local redis = require "resty.iredis"
2
3 local function get_from_redis(key)
4     local red = redis:new({"ip"]="127.0.0.1", ["port"]=6379})
5     local res, err = red:get(key)
6     return res
7 end
8
9 local function set_to_cache(key, value, exptime)
10     if not exptime then
11         exptime = 0
12     end
13     local cache ngx = ngx.shared.cache_ngx
14     local succ, err, forcible = cache_ngx:set(key, value, exptime)
15     return succ
16 end
17
18 local function get_from_cache(key)
19     local cache_ngx = ngx.shared.cache_ngx
20     local value = cache_ngx:get(key)
21     if not value then
22         local lock = require "resty.lock"
23         local lock = lock:new("my_locks")
24         lock:lock("my_key")
25         value = get_from_redis(key)
26         lock:unlock()
27
28         set_to_cache(key, value, 100)
29     end
30     return value
31 end
32
33 local res = get_from_cache('ad0001')
34 ngx.say(res)
35
iredis_get3.lua
```

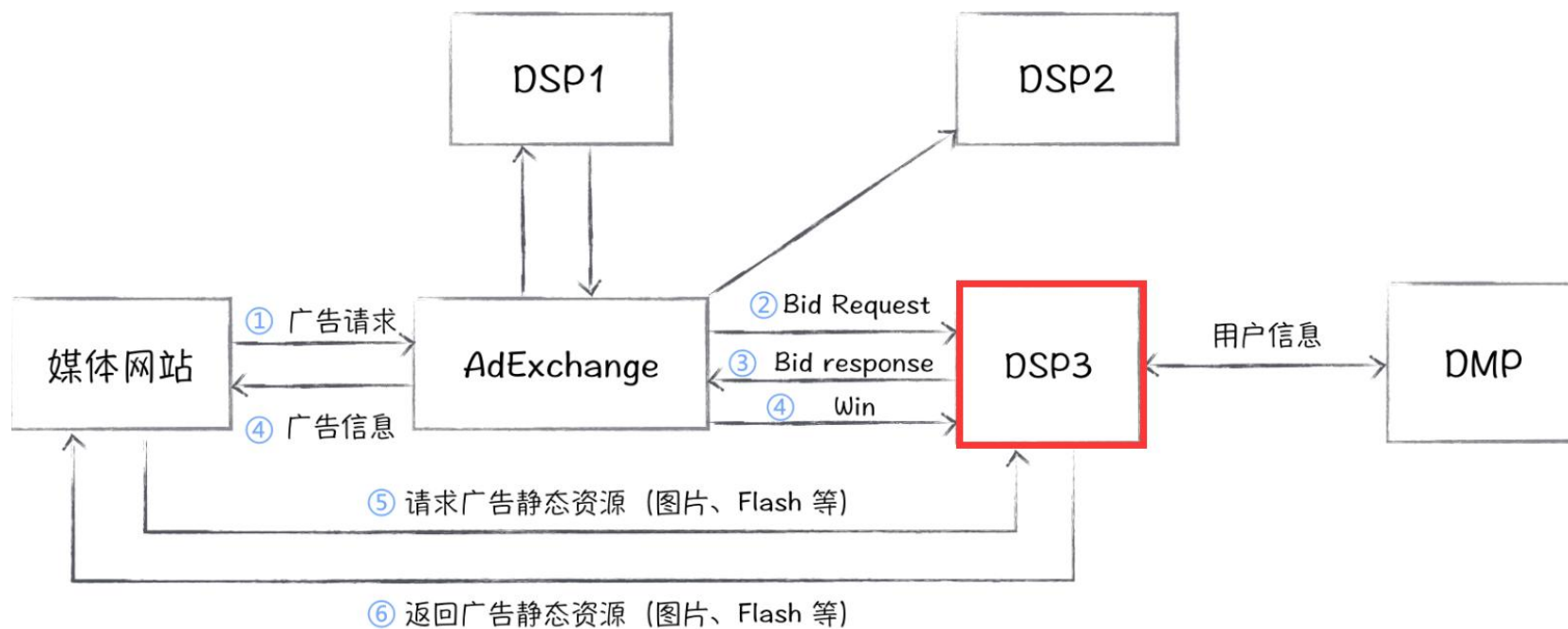
E 案例分享--RTB

案例分享--RTB



RTB

RTB 广告投放流程详解



案例分享--RTB



DSP处理流程



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



Thank
you