

NGINX, Lua, and beyond

😊 *agentzh@gmail.com* 😊
Yichun Zhang (@agentzh)



2014.02.25

ngx_echo

```
echo "hello world";
```

```
echo_after_body "some footer...";
```

```
echo_duplicate 1024 "a";
```

```
echo_sleep 1.5;
```

```
echo_exec /foo;
```

```
echo_location /baz;
```

```
echo_subrequest_async DELETE /bar
```

```
$echo_request_method
```

ngx_headers_more

more_clear_headers Server X-Powered-By;

more_set_headers 'Foo: bar';

more_set_input_headers 'User-Agent: curl';

ngx_set_misc

```
set_unescape_uri $name $arg_name;
```

```
set_if_empty $name "anonymous";
```

```
set_random $rand 1 10;
```

```
set_formatted_local_time $timestr "%a %b %e %H:%M:%S %Y %Z";
```

NGINX

ngx-memc




```
location = /memc-set {  
    set $memc_cmd set;  
    set $memc_key foo;  
    set $memc_value blah;  
    set $memc_exptime 60; # 60 sec  
    memc_pass 127.0.0.1:11211;  
}
```

Ragel

ngx_redis2

```
location = /redis-test {  
    redis2_query del key1;  
    redis2_query lpush key1 A;  
    redis2_query lpush key1 B;  
    redis2_pass 127.0.0.1:6379;  
}
```

NGINX

ngx_drizzle

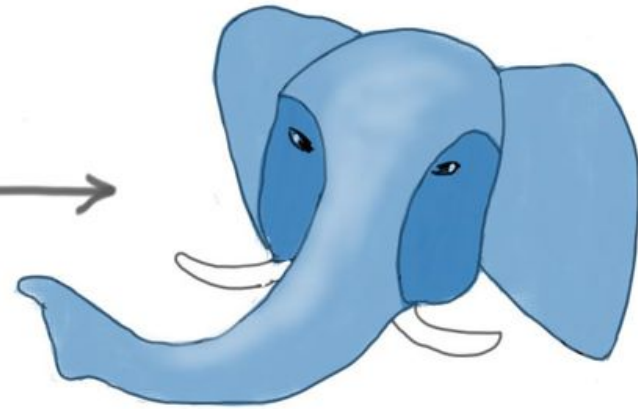
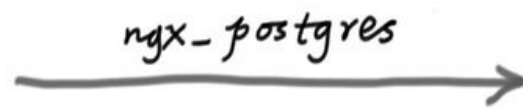


```
upstream mysql_backend {  
    drizzle_server 127.0.0.1:3306 dbname=test  
    password=some_pass user=monty protocol=mysql;  
}
```

```
server {  
    location = /mysql {  
        set $query 'select * from cats';  
        drizzle_query $query;  
        drizzle_pass mysql_backend;  
        rds_json on; # or rds_csv on  
    }  
}
```

libdrizzle

NGINX



libpq

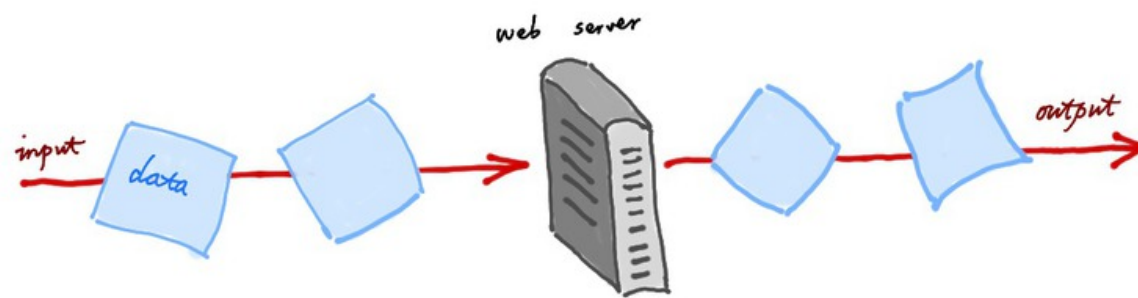
Oracle? OCI?

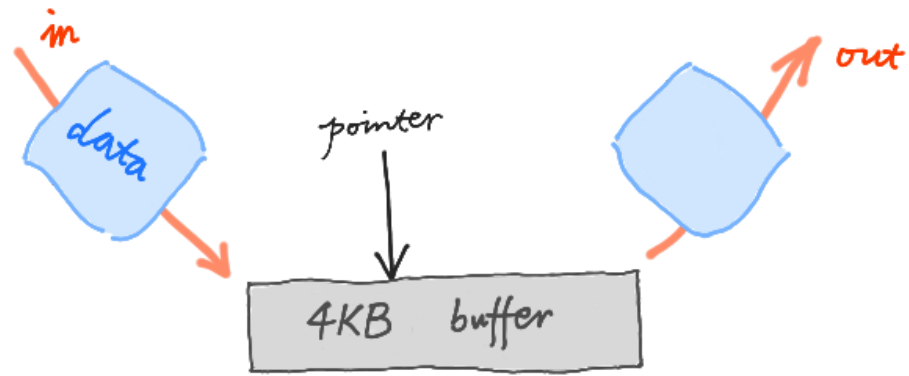
ngx_srcache

```
location /foo {  
    set $key $uri$args;  
  
    srcache_fetch GET /memc $key;  
    srcache_store PUT /memc $key;  
  
    srcache_store_statuses 200 301 302;  
  
    # proxy_pass/fastcgi_pass/drizzle_pass/echo/etc  
}
```

ngx_replace_filter

```
location ~ '\.cpp$' {  
    # proxy_pass/fastcgi_pass/root/...  
    replace_filter_types text/plain;  
  
    # skip C/C++ string literals:  
    replace_filter "'(?:\\\[^\n]|^[']\n)*)" $& g;  
    replace_filter '"(?:\\\[^\n]|^["]\n)*)" $& g;  
  
    # remove all those ugly C/C++ comments:  
    replace_filter '/\*.*?\*/|//[^\n]*' '' g;  
}
```





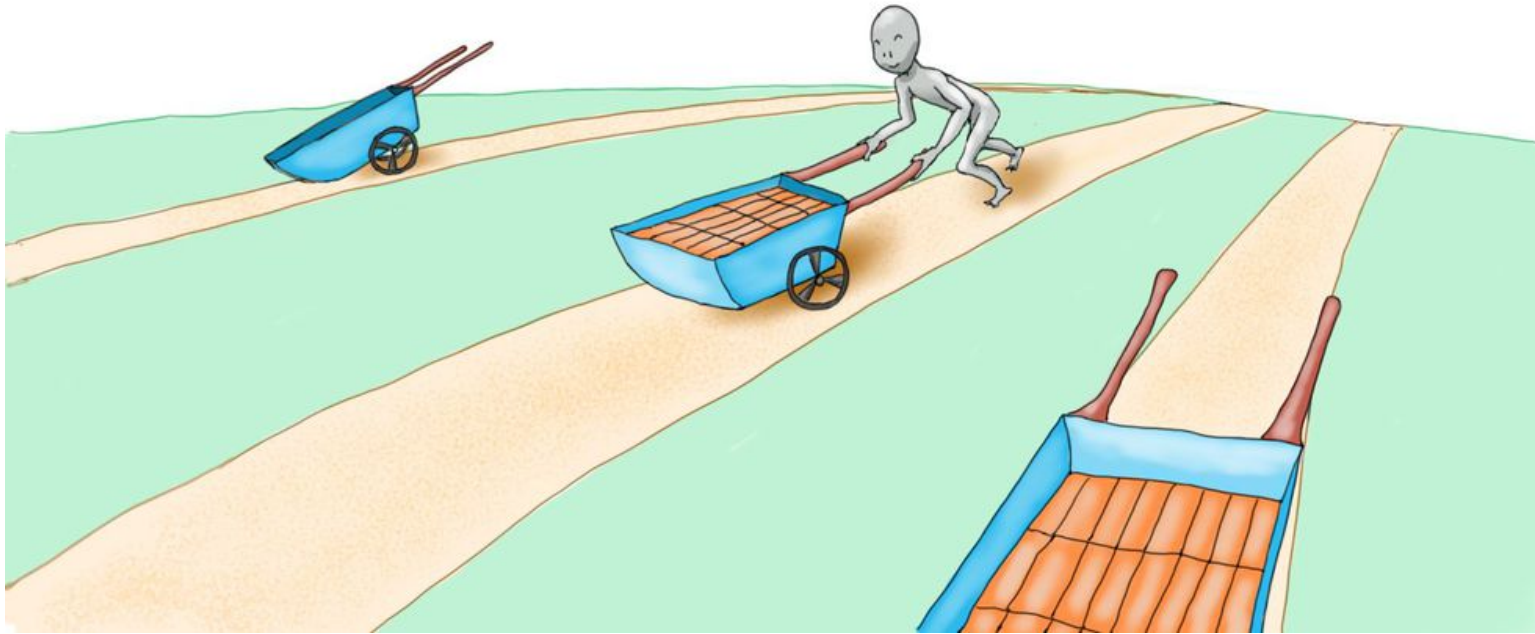
libsregex

nginx.conf *scripting*

ngx_lua

init_by_lua
init_worker_by_lua
set_by_lua
rewrite_by_lua
access_by_lua
content_by_lua
header_filter_by_lua
body_filter_by_lua
log_by_lua

"Light threads" based on
Lua *coroutines*



```
ngx.location.capture("/foo")
```

```
ngx.location.capture_multi{ {"/foo"}, {"/bar"} }
```

```
local sock = ngx.socket.tcp()  
local ok, err = sock:connect("127.0.0.1", 1234)  
local bytes, err = sock:send("hello")  
local data, err = sock:receive("*1")
```



```
local sock = ngx.socket.udp()  
local ok, err = sock:setpeer("127.0.0.1", 5432)  
local bytes, err = sock:send("my query")  
local dgram, err = sock:receive()
```

```
local req_body_sock = ngx.req.socket()  
local raw_sock = ngx.req.socket(true)
```

```
local t = ngx.thread.spawn(func)  
local ok, res = ngx.thread.wait(t)
```

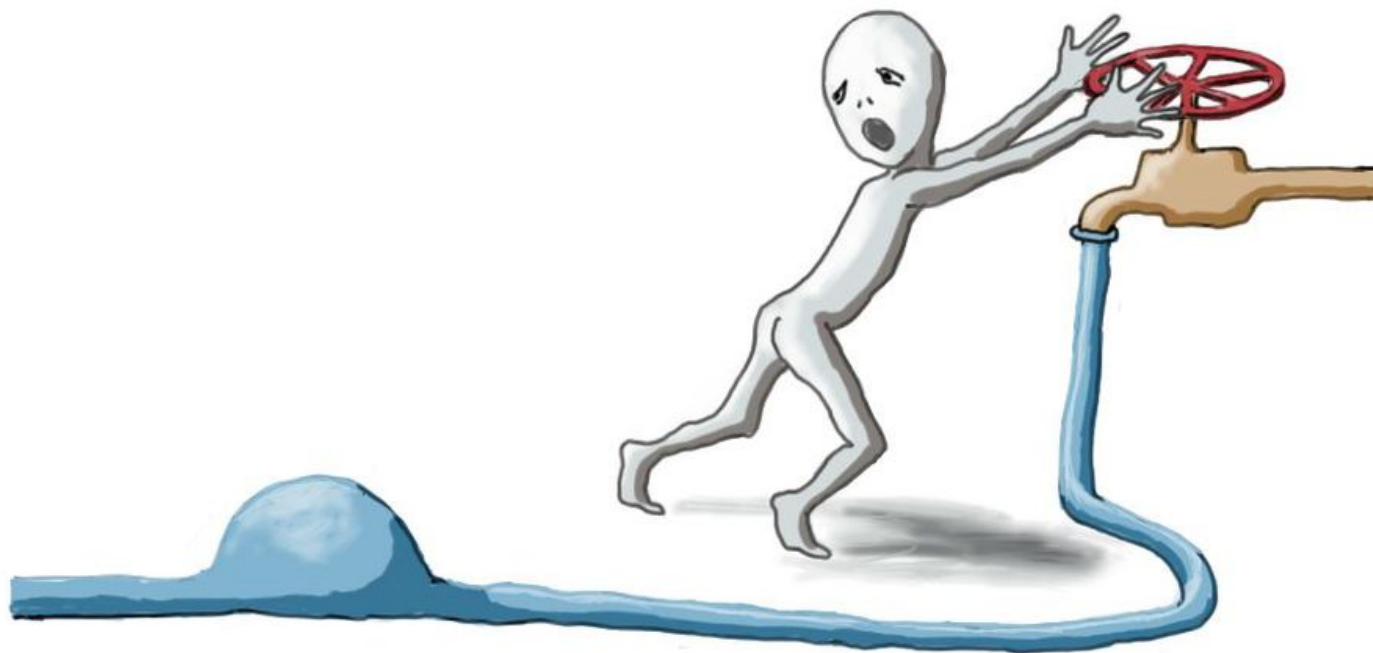
```
local dogs = ngx.shared.dogs  
dogs:add("Tom", 5)  
dogs:set("Tom", 7)  
dogs:incr("Tom", 1)  
dogs:delete("Tom", 1)
```

```
local function do_job()
```

```
    . . .
```

```
end
```

```
local ok, err = ngx.timer.at(1.5, do_job)
```



```
while true do
    local chunk, err =
        read_data_chunk_from_backend()
    if not chunk then
        -- error and eof handling
    end

    local ok, err = ngx.print(chunk)
    -- handle errors here if any

    local ok, err = ngx.flush(true)
    -- handle errors here if any
end
```

lua-resty-memcached

lua-resty-redis

lua-resty-mysql

lua-resty-dns

lua-resty-upload

lua-resty-websocket

lua-resty-lock

lua-resty-upstream-healthcheck

LuaJIT

OpenResty

CloudFlare Lua **CDN**

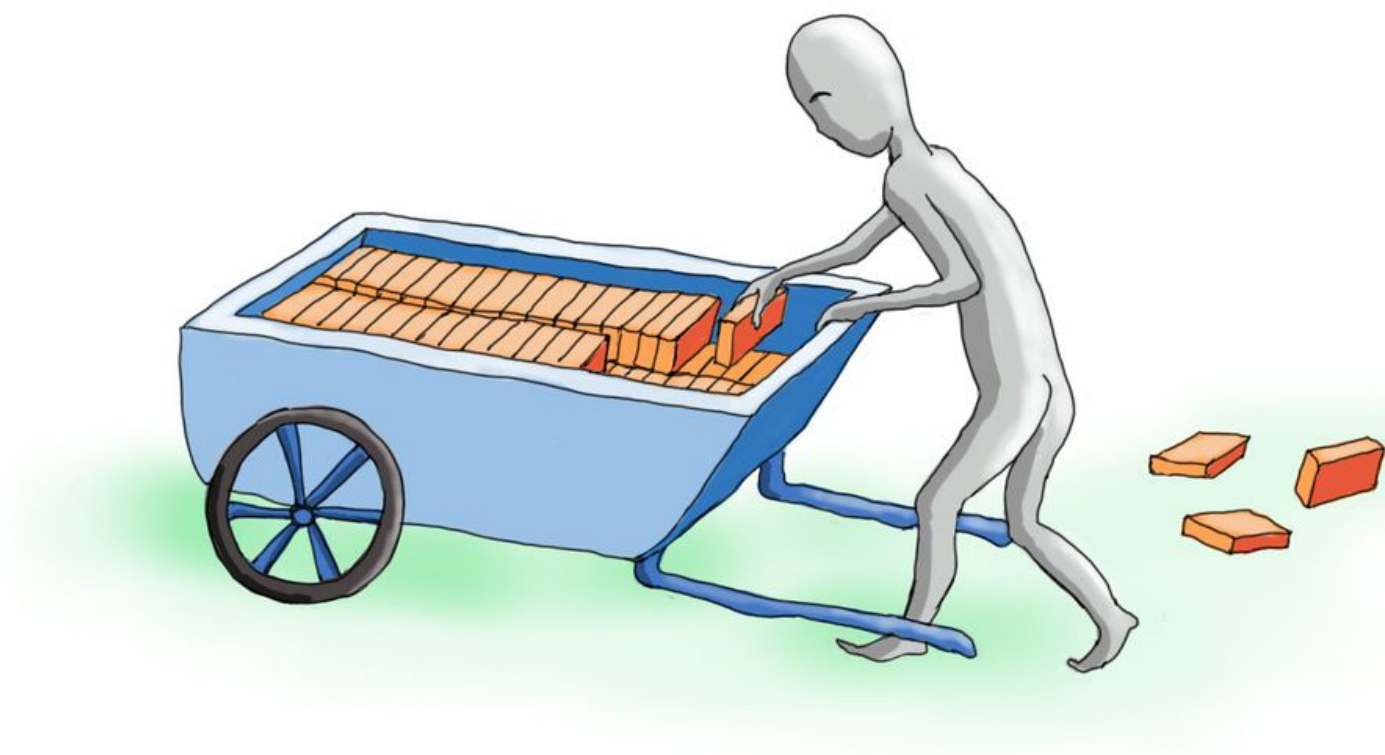


CloudFlare Lua WAF



Testing

No-pool patch for NGINX



mockeagain



```
export MOCKEAGAIN=r
```

```
export LD_PRELOAD=/path/to/mockagain.so
```

```
# edit nginx.conf to add "env" directives
```

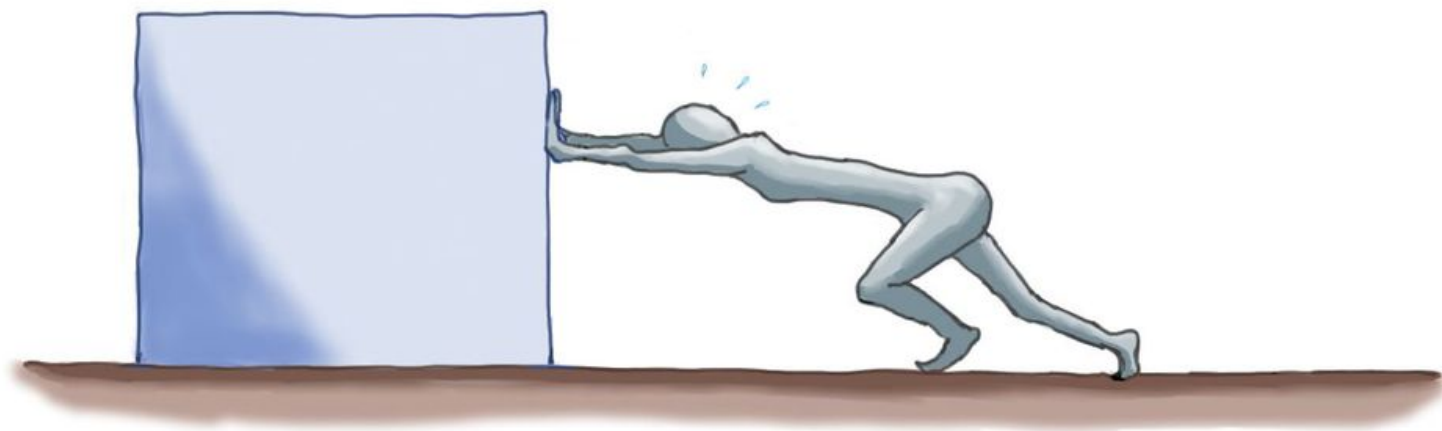
```
# start nginx and test
```

nginx.conf

env MOCKEAGAIN;

env LD_PRELOAD;

Better than the TCP proxy approach



```
export  MOCKEAGAIN=w
```

Mocking socket write timeout

```
export MOCKEAGAIN=w
```

```
export MOCKEAGAIN_WRITE_TIMEOUT_PATTERN='hello world'
```


Test::Nginx

```
=== TEST 1: echo hello world
--- config
    location = /t {
        echo "hello world";
    }
--- request
GET /t
--- response_body
hello world
```

--- more_headers

User-Agent: Opera/9.80 (Macintosh; Intel Mac OS X 10.7.4; U; en)

--- stap

```
F(ngx_http_core_content_phase) {  
    printf("content: opera: %d\n", $r->headers_in->opera)  
}
```

--- stap_out

content: opera: 0

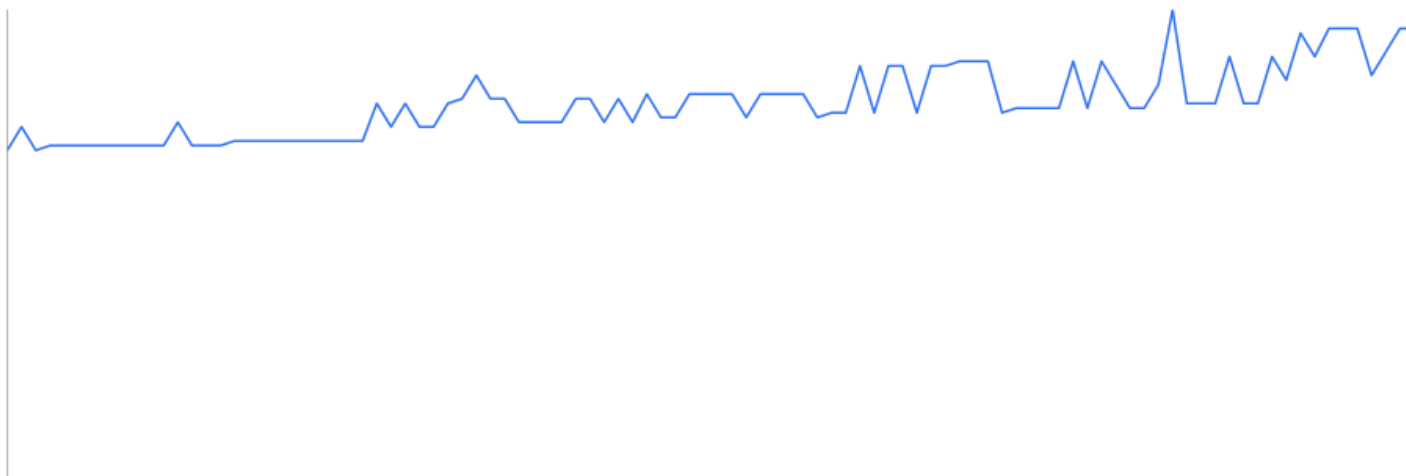
LeakTest: k=10.5

Up-Edges: 52

Stable point: N/A

RSS Samples (in KB):

3596	3860	3604	3608	3612	3616	3620	3624	3628
3628	3632	3636	3904	3648	3652	3656	3660	3668
3672	3676	3680	3684	3688	3692	3696	3700	4096
3844	4104	3848	3852	4116	4120	4388	4128	4132
3876	3880	3884	3888	4156	4160	3904	4168	3912
4176	3920	3928	4192	4192	4196	4200	3944	4208
4212	4216	4220	3964	3968	3972	4496	3980	4508
4512	3992	4520	4524	4528	4532	4536	4016	4020
4024	4028	4032	4560	4044	4568	4312	4056	4060
4320	5116	4068	4072	4076	4604	4084	4088	4620
4356	4884	4628	4892	4900	4900	4384	4648	4916



Test Cluster on Amazon EC2

opsboy

```
$ ./dispatcher -r -t 170 -a 'linux x86_64' ngx_lua
```

Requires at least 5 machines.

bucket 1: tl-ngx_lua force=1 (242 min)

**bucket 2: twv-ngx_lua tw-ngx_lua to-
ngx_lua force=1 (160 min)**

bucket 3: trv-ngx_lua tr-ngx_lua force=1 (125 min)

bucket 4: tv-ngx_lua t-ngx_lua force=1 (118 min)

bucket 5: thv-ngx_lua th-ngx_lua force=1 (74 min)

qa.openresty.org

(This page was automatically generated by the tools in the [opsboy](#) project on 20:08:16 05-Feb-2014 GMT.)

Amazon EC2 Test Cluster Reports for Nginx and its Components

- [Reports](#)
 - [Linux x86_64](#)
 - [nginx/1.5.9 \(no pool\)](#)
 - [nginx/1.5.8 \(no pool\)](#)
 - [miscellaneous](#)
 - [Linux i386](#)
 - [nginx/1.5.9 \(no pool\)](#)
 - [nginx/1.5.8 \(no pool\)](#)
 - [miscellaneous](#)
- [Archive](#)
- [Maintainer](#)

Reports

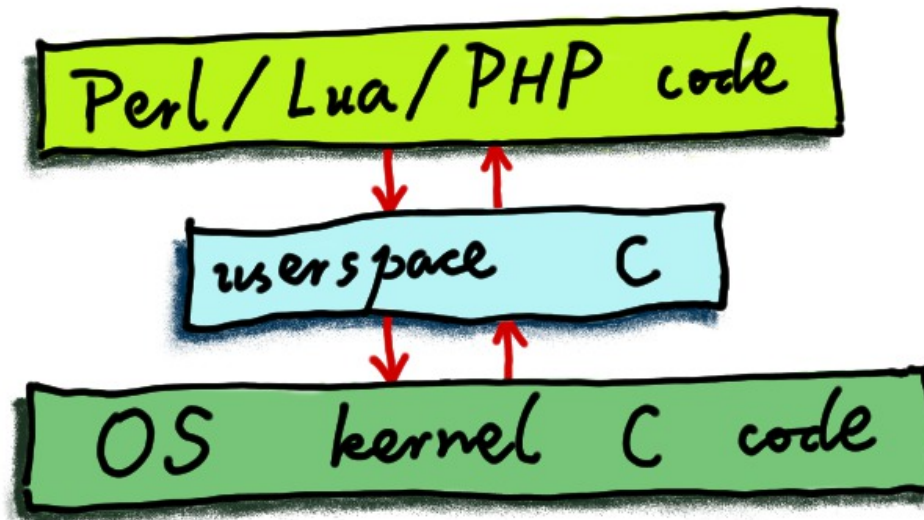
Linux x86_64

nginx/1.5.9 ([no pool](#))

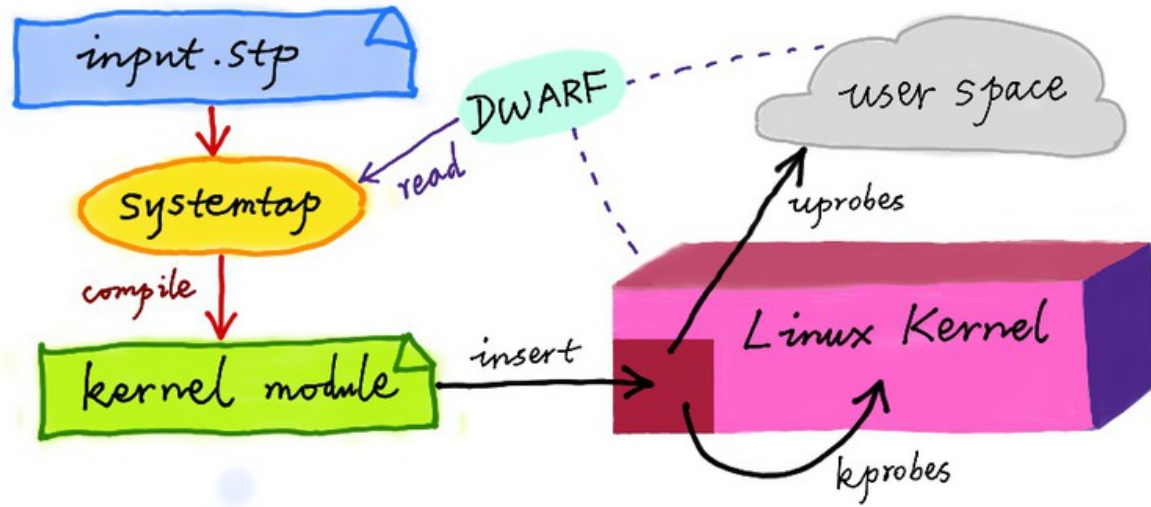
Component	plain	valgrind/memcheck	mockeagain (R)	mockeagain (R) valgrind/memcheck
lua-resty-core	PASS NEW	PASS NEW	PASS NEW	PASS NEW
lua-resty-dns	PASS	PASS	PASS	PASS
lua-resty-lock	PASS	PASS	PASS	PASS
lua-resty-logger-socket	PASS	PASS	PASS	PASS
lua-resty-memcached	PASS	PASS	PASS	PASS

Online debugging and profiling

systemtap



The Software Stack



nginx-systemtap-toolkit
stapxx (stap++)

```
$ ./ngx-req-distr -c -  
m `cat /opt/nginx/logs/nginx.pid`  
Tracing 4394 4395 4396 4397 (/opt/nginx/sbin/nginx)...  
Hit Ctrl-C to end.  
^C  
worker 4394:      0 reqs,      0 conns  
worker 4395:    2100 reqs,    21 conns  
worker 4396:    501 reqs,     6 conns  
worker 4397:    2100 reqs,    21 conns
```


show the details of the shm zone named "dogs"

\$./ngx-shm -p 15218 -n dogs

Tracing 15218 (/opt/nginx/sbin/nginx)...

shm zone "dogs"

owner: ngx_http_lua_shdict

total size: 100 KB

free pages: 88 KB (22 pages, 1 blocks)

22 microseconds elapsed in the probe.

```
$ ngx-rps.sxx -x 19647
```

```
WARNING: Tracing process 19647.
```

```
Hit Ctrl-C to end.
```

```
[1376939543] 300 req/sec
```

```
[1376939544] 235 req/sec
```

```
[1376939545] 235 req/sec
```

```
[1376939546] 166 req/sec
```

```
$ epoll-loop-blocking-distr.sxx -x 19647 --arg time=60
```

```
Start tracing 19647...
```

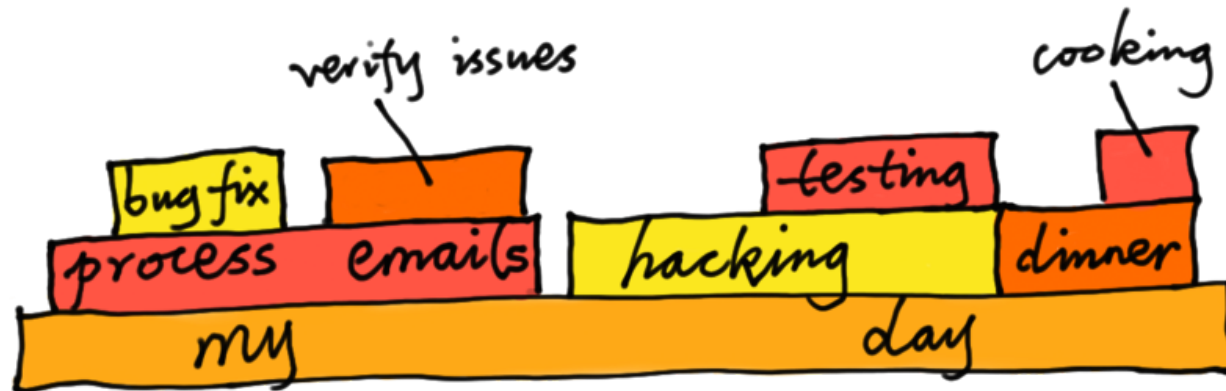
```
Please wait for 60 seconds.
```

```
Distribution of epoll loop blocking latencies (in milliseconds)
```

```
max/avg/min: 1097/0/0
```

value	-----	count
0	@@	18471
1	@@@@@@@@@@	3273
2	@	473
4		119
8		67
16		51
32		35
64		20
128		23
256		9
512		2
1024		2
2048		0

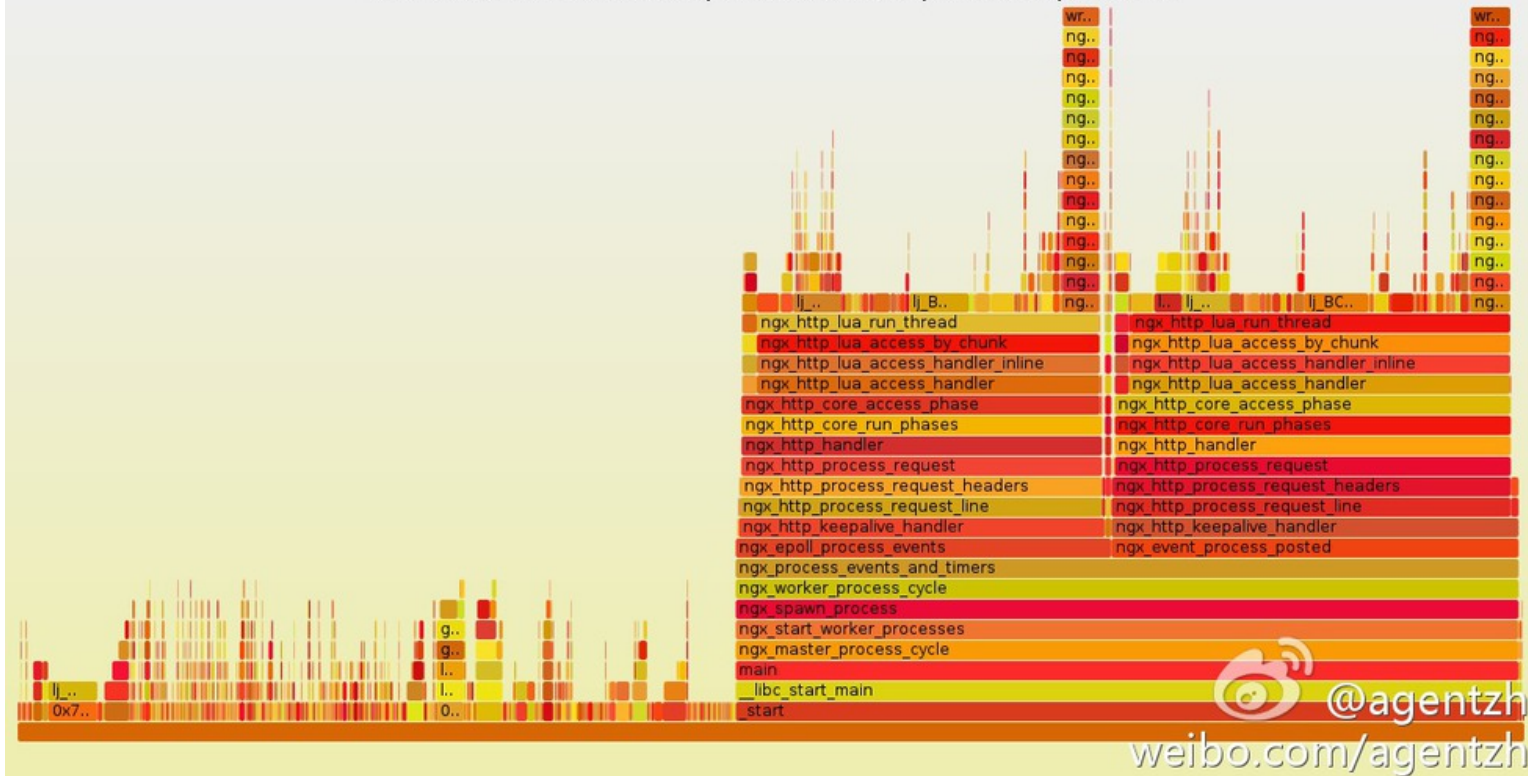
Flame Graphs

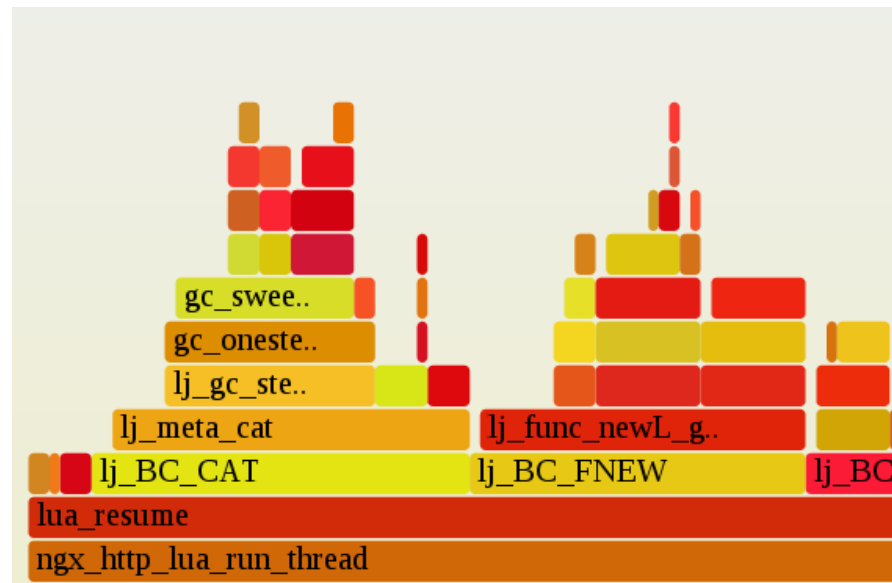


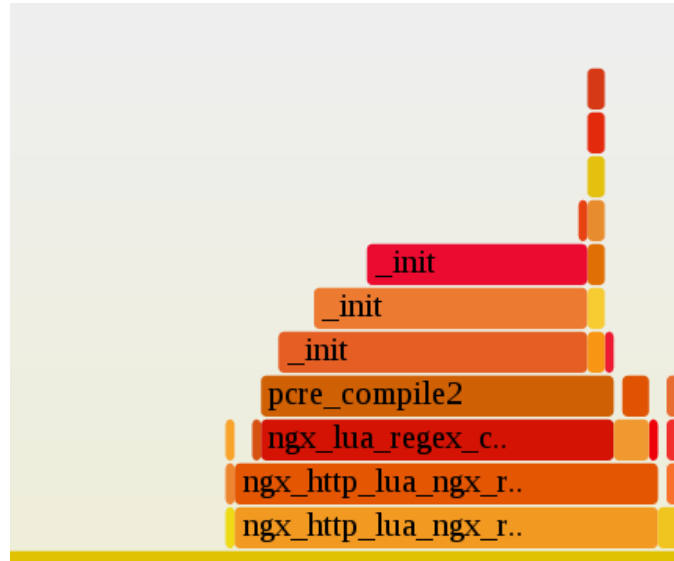
Flame Graph for My Day

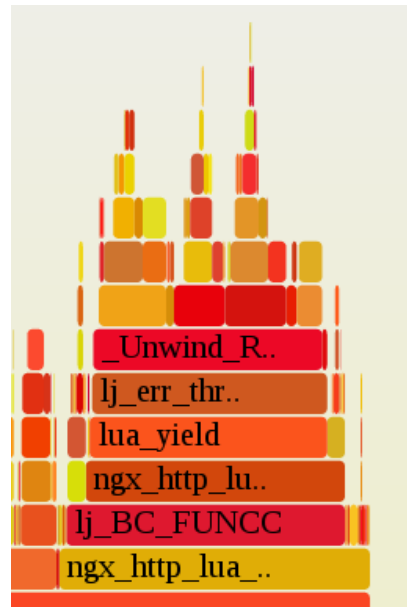
on-CPU Flame Graphs

C-Land on-CPU Time Flame Graph for Lua WAF (simplest GET requests) - V3









on-CPU Flame Graph for the KTF5 client

The flame graph visualizes the execution of the KTF5 client on the CPU. The horizontal axis represents time, and the vertical axis represents the call stack. The graph shows a sequence of function calls and returns, with colors indicating different functions. The functions listed on the right side of the graph are: nss_hosts_..., std::basic_s..., kyototycoon::..., kyototycoon::..., kyotocabinet::..., kyototycoon::b..., kyototycoon::tsv..., kyototycoon::ts..., kyototycoon::RPCClient::call(std::b..., kyototycoon::RemoteDB::get(char con..., kyototycoon::RemoteDB::get(std::bas..., kyototycoon::Release(char const*, ...), KyototycoonFS::Read(char const*, char*, unsigned long, long..), wrap_release, fuse_destroy, fuse_notify_poll, fuse_reply_err, fuse_session_loop, start_thread, and _clone. The graph shows a complex sequence of calls and returns, with some functions being called multiple times. The colors of the bars represent different functions, and the length of the bars indicates the duration of the function execution.

Function:

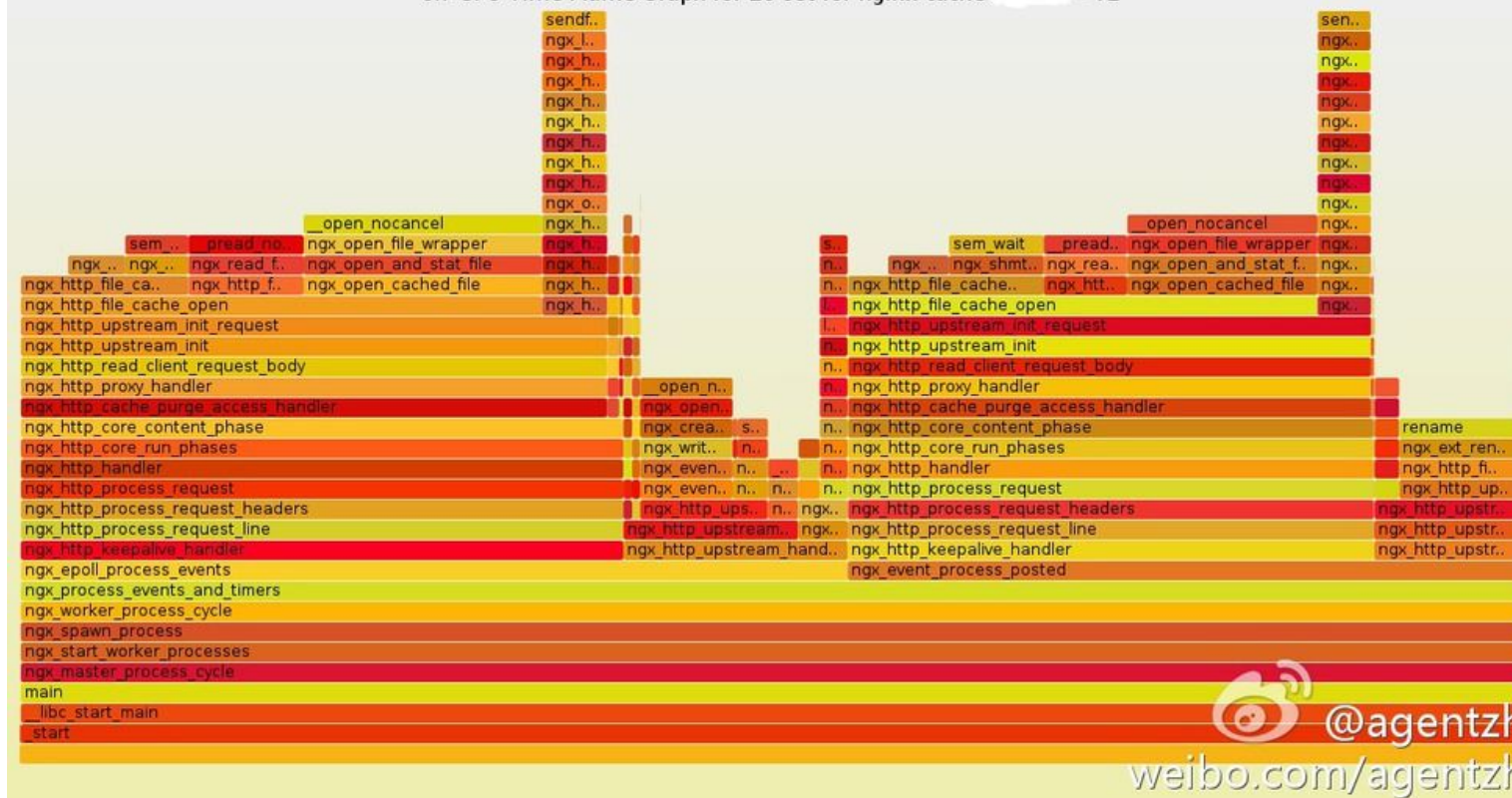
Function:

[illegible]

Function:

off-CPU Flame Graphs

- V2



- V4

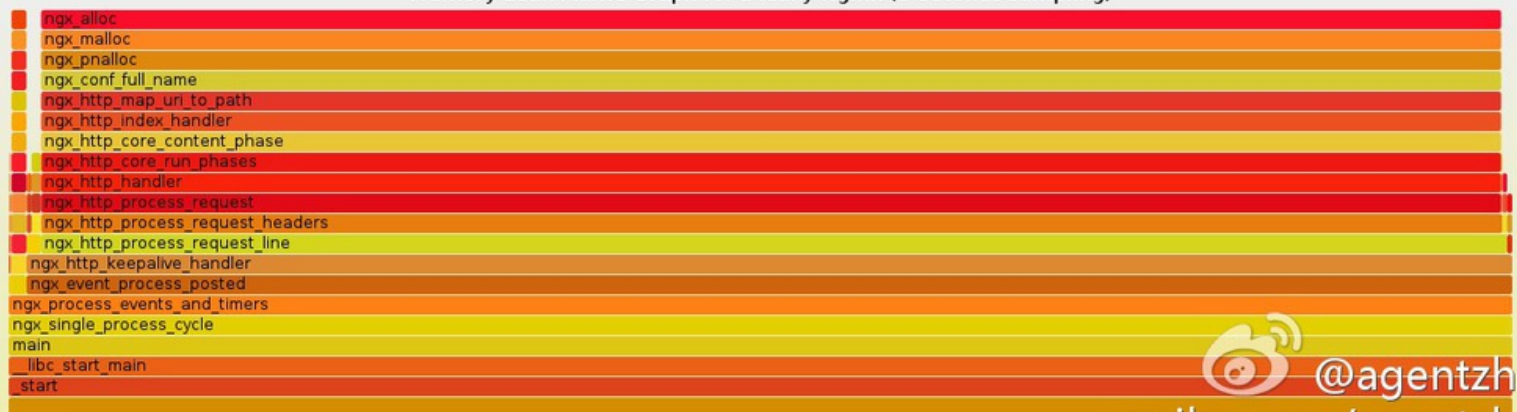


[illegible]

Function:

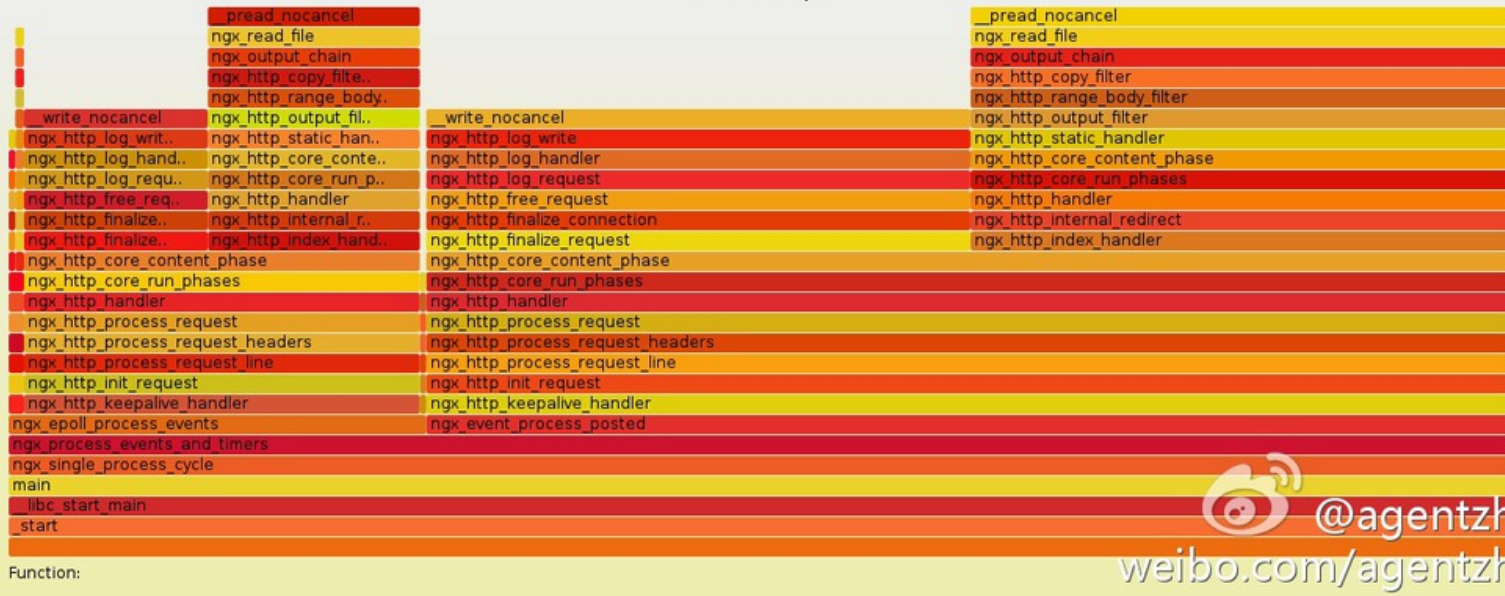
Memory Leak Flame Graphs

Memory Leak Flame Graph for a leaky Nginx (5 seconds sampling)



File IO Flame Graphs

Flame Graph



DWARF debug info format





nginx-gdb-utils

(gdb) source luajit21.py

(gdb) lvmst

current VM state: C code from interpreted Lua

(gdb) lbt

builtin#166

builtin#195

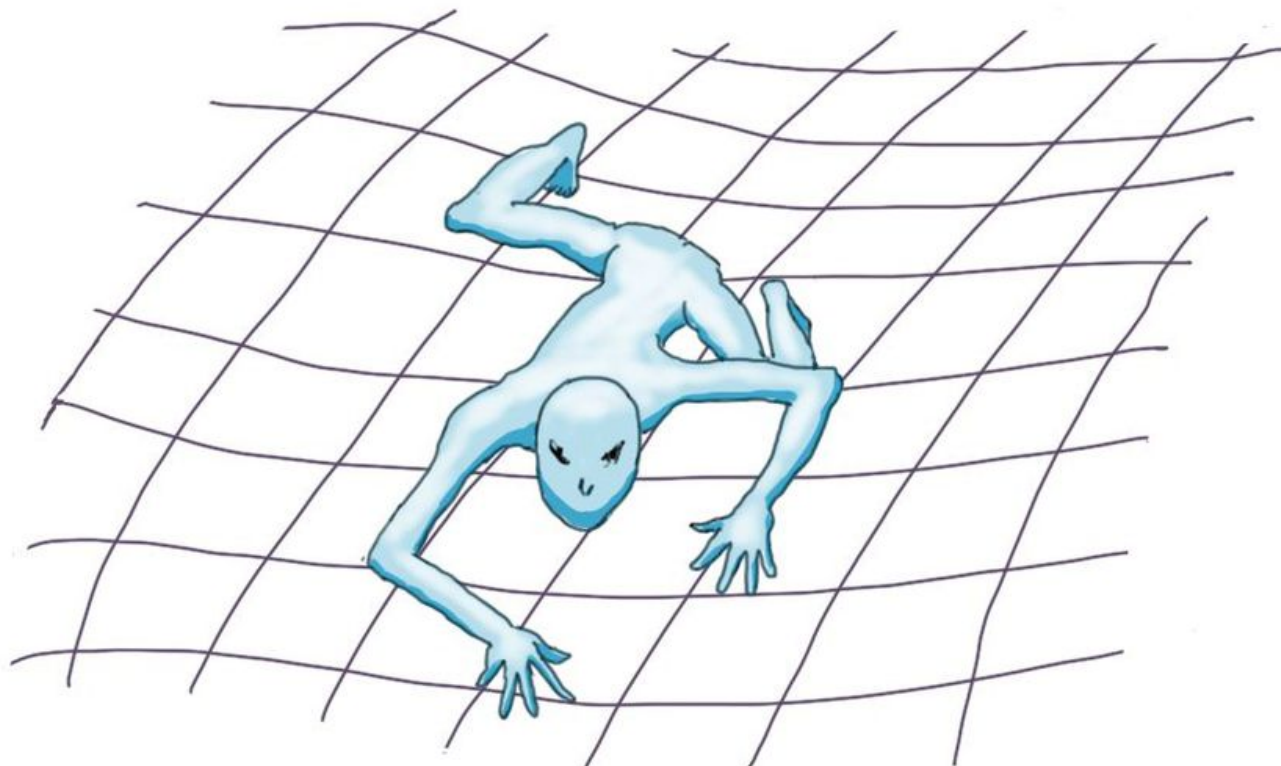
builtin#187

**@/home/agentzh/git/lua-resty-
core/lib/resty/core/regex.lua:588**

content_by_lua:10

```
(gdb) source ngx-raw-req.py  
(gdb) ngx-raw-req r  
GET /foo HTTP/1.0  
Host: bar.com  
User-Agent: curl  
Accept-Encoding: gzip
```

Spider binary data structures in the process space



We are hiring :)

😊 *Any questions?* 😊

