OpenResty 项目性能优化实践

Alex Zhang

Github: https://github.com/tokers

2017/12/23

OpenResty Meetup 杭州

又拍云

- 常用性能分析工具
- 基于 **OpenResty** 的项目的特点
- 基于 **OpenResty** 的项目细节优化

常用性能分析工具

资源分析

- top
- pidstat
- iostat
- …

负载分析

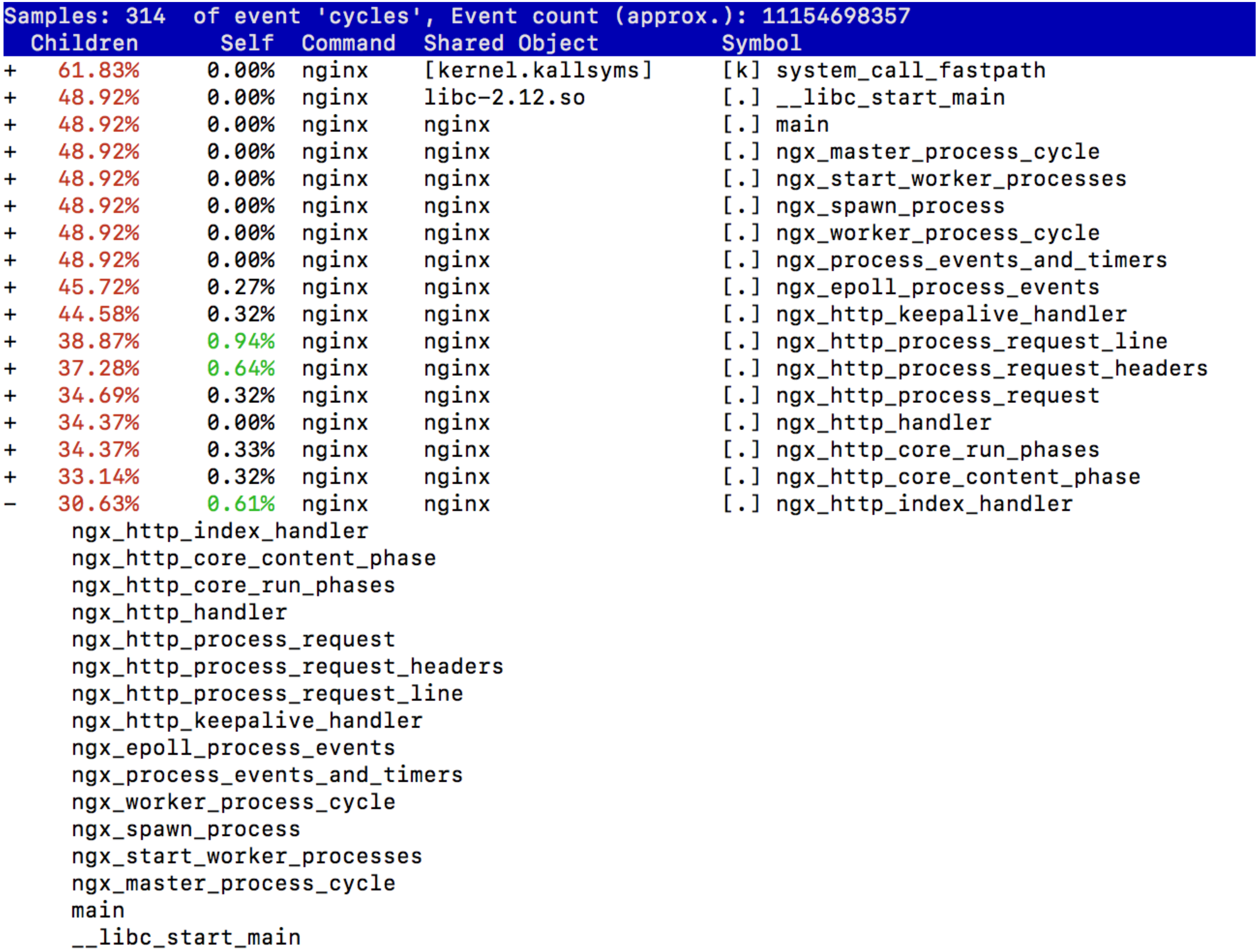
- perf
- SystemTap
- FlameGraph
- …

Perf

- http://www.brendangregg.com/perf.html
- 多种不同种类事件， **perf list**
- **进程级别**的事件统计, **perf stat -p <pid>**
- **函数级别**的事件统计， **perf report -p <pid> && perf record**

```
# perf stat -e 'context-switches,page-faults,branch-misses' -p 2623564
^C
 Performance counter stats for process id '2623564':

                 3      context-switches
                 0      page-faults
             1,346      branch-misses

       3.223158849 seconds time elapsed
```

```
   # perf record -F 100 -p 2623564 -g -- sleep 5
   [ perf record: Woken up 1 times to write data ]
   [ perf record: Captured and wrote 0.009 MB perf.data (5 samples) ]

   # perf report
```



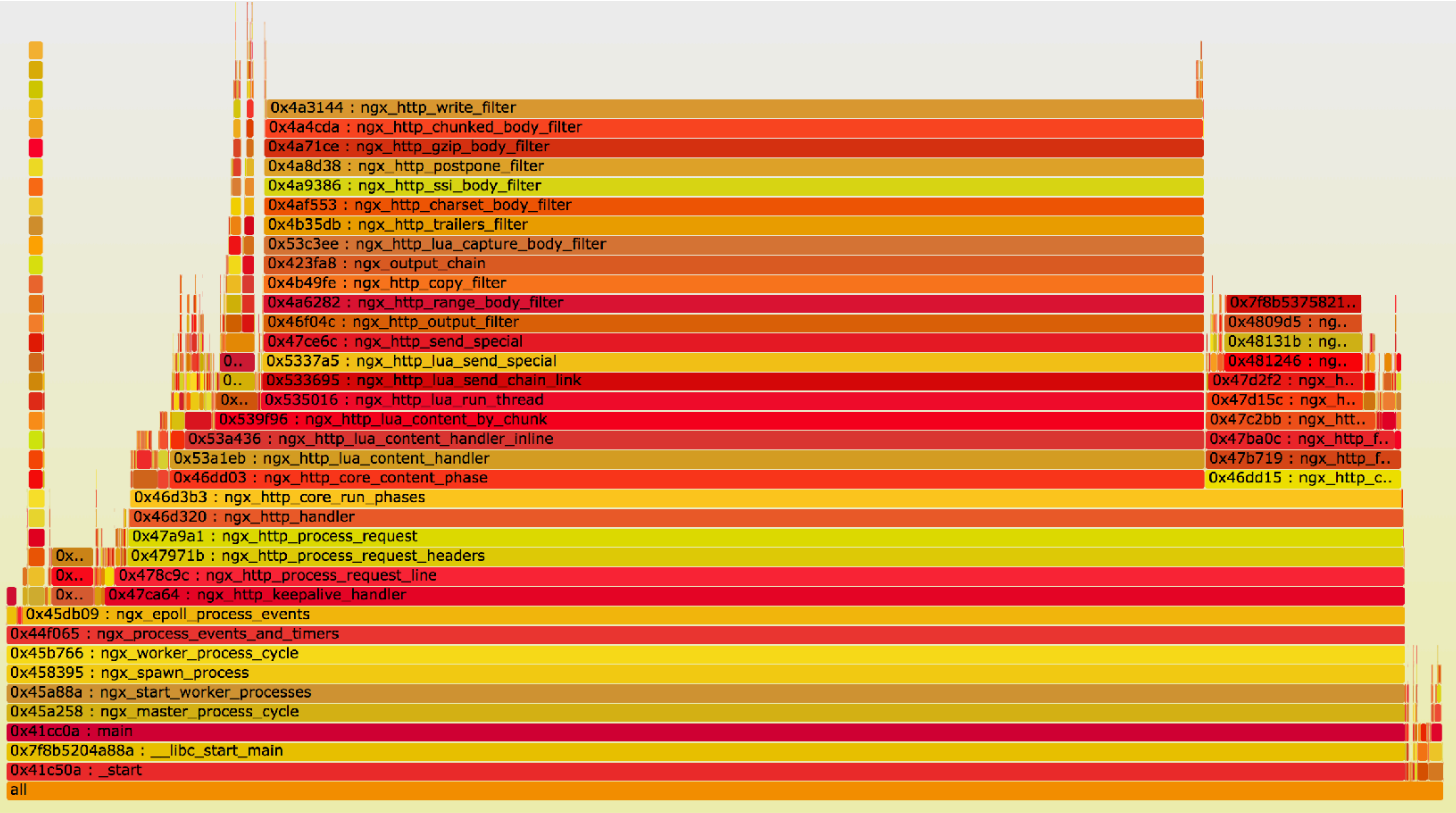
```
Samples: 314  of event 'cycles', Event count (approx.): 11154698357
  Children      Self  Command  Shared Object          Symbol
+   61.83%     0.00%  nginx    [kernel.kallsyms]      [k] system_call_fastpath
+   48.92%     0.00%  nginx    libc-2.12.so           [.] __libc_start_main
+   48.92%     0.00%  nginx    nginx                  [.] main
+   48.92%     0.00%  nginx    nginx                  [.] ngx_master_process_cycle
+   48.92%     0.00%  nginx    nginx                  [.] ngx_start_worker_processes
+   48.92%     0.00%  nginx    nginx                  [.] ngx_spawn_process
+   48.92%     0.00%  nginx    nginx                  [.] ngx_worker_process_cycle
+   48.92%     0.00%  nginx    nginx                  [.] ngx_process_events_and_timers
+   45.72%     0.27%  nginx    nginx                  [.] ngx_epoll_process_events
+   44.58%     0.32%  nginx    nginx                  [.] ngx_http_keepalive_handler
+   38.87%     0.94%  nginx    nginx                  [.] ngx_http_process_request_line
+   37.28%     0.64%  nginx    nginx                  [.] ngx_http_process_request_headers
+   34.69%     0.32%  nginx    nginx                  [.] ngx_http_process_request
+   34.37%     0.00%  nginx    nginx                  [.] ngx_http_handler
+   34.37%     0.33%  nginx    nginx                  [.] ngx_http_core_run_phases
+   33.14%     0.32%  nginx    nginx                  [.] ngx_http_core_content_phase
-   30.63%     0.61%  nginx    nginx                  [.] ngx_http_index_handler
     ngx_http_index_handler
     ngx_http_core_content_phase
     ngx_http_core_run_phases
     ngx_http_handler
     ngx_http_process_request
     ngx_http_process_request_headers
     ngx_http_process_request_line
     ngx_http_keepalive_handler
     ngx_epoll_process_events
     ngx_process_events_and_timers
     ngx_worker_process_cycle
     ngx_spawn_process
     ngx_start_worker_processes
     ngx_master_process_cycle
     main
     __libc_start_main
```

SystemTap

- **动态追踪** -自定义探针
- **DSL** - 简单灵活的脚本语言
- **用户态**空间追踪和**内核态**空间追踪
- **调用栈回溯**
- **非侵入式**

```
global connections
global counts

probe begin {
        warn(sprintf("Tracing /usr/local/nginx/sbin/nginx...\n"))
}

probe process("/usr/local/openresty/nginx/sbin/nginx").function("ngx_process_events_and_timers") {
        type = @var("ngx_process@ngx_process_cycle.c")
        if (type == 0 || type == 3) {
                connection_n = @cast($cycle, "ngx_cycle_t")->connection_n
                free_connection_n = @cast($cycle, "ngx_cycle_t")->free_connection_n
                connections[pid()] <<< (connection_n - free_connection_n)
                counts[pid()] <<< 1
        }
}

probe timer.s(5) {
        printf("Time's up. Quitting now...(it may take a while)\\n\\n")
        exit()
}

probe end {
        foreach (pid in connections) {
                connection = @count(connections[pid])
                count = @count(counts[pid])
                avg = connection / count
                printf("pid: %d, average connections: %d\n", pid, avg)
        }
}
```

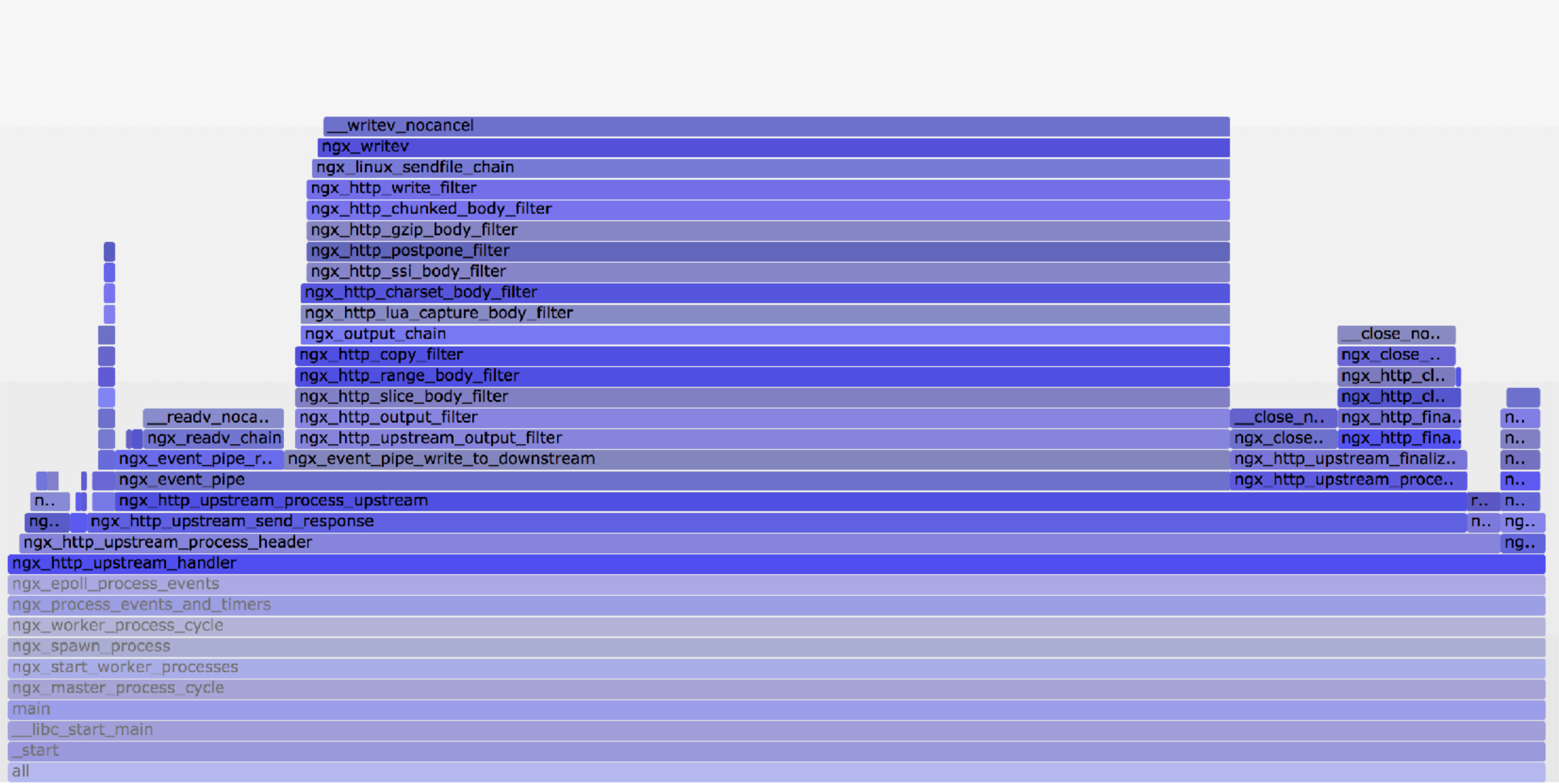
FlameGraph


0x4a3144 : ngx_http_write_filter
0x4a4cda : ngx_http_chunked_body_filter
0x4a71ce : ngx_http_gzip_body_filter
0x4a8d38 : ngx_http_postpone_filter
0x4a9386 : ngx_http_ssi_body_filter
0x4af553 : ngx_http_charset_body_filter
0x4b35db : ngx_http_trailers_filter
0x53c3ee : ngx_http_lua_capture_body_filter
0x423fa8 : ngx_output_chain
0x4b49fe : ngx_http_copy_filter
0x4a6282 : ngx_http_range_body_filter
0x7f8b5375821..
0x46f04c : ngx_http_output_filter
0x4809d5 : ng..
0x47ce6c : ngx_http_send_special
0x48131b : ng..
0..
0x5337a5 : ngx_http_lua_send_special
0x481246 : ng..
0..
0x533695 : ngx_http_lua_send_chain_link
0x47d2f2 : ngx_h..
0x..
0x535016 : ngx_http_lua_run_thread
0x47d15c : ngx_h..
0x539f96 : ngx_http_lua_content_by_chunk
0x47c2bb : ngx_htt..
0x53a436 : ngx_http_lua_content_handler_inline
0x47ba0c : ngx_http_f..
0x53a1eb : ngx_http_lua_content_handler
0x47b719 : ngx_http_f..
0x46dd03 : ngx_http_core_content_phase
0x46dd15 : ngx_http_c..
0x46d3b3 : ngx_http_core_run_phases
0x46d320 : ngx_http_handler
0x47a9a1 : ngx_http_process_request
0x..
0x47971b : ngx_http_process_request_headers
0x..
0x478c9c : ngx_http_process_request_line
0x..
0x47ca64 : ngx_http_keepalive_handler
0x45db09 : ngx_epoll_process_events
0x44f065 : ngx_process_events_and_timers
0x45b766 : ngx_worker_process_cycle
0x458395 : ngx_spawn_process
0x45a88a : ngx_start_worker_processes
0x45a258 : ngx_master_process_cycle
0x41cc0a : main
0x7f8b5204a88a : __libc_start_main
0x41c50a : _start
all

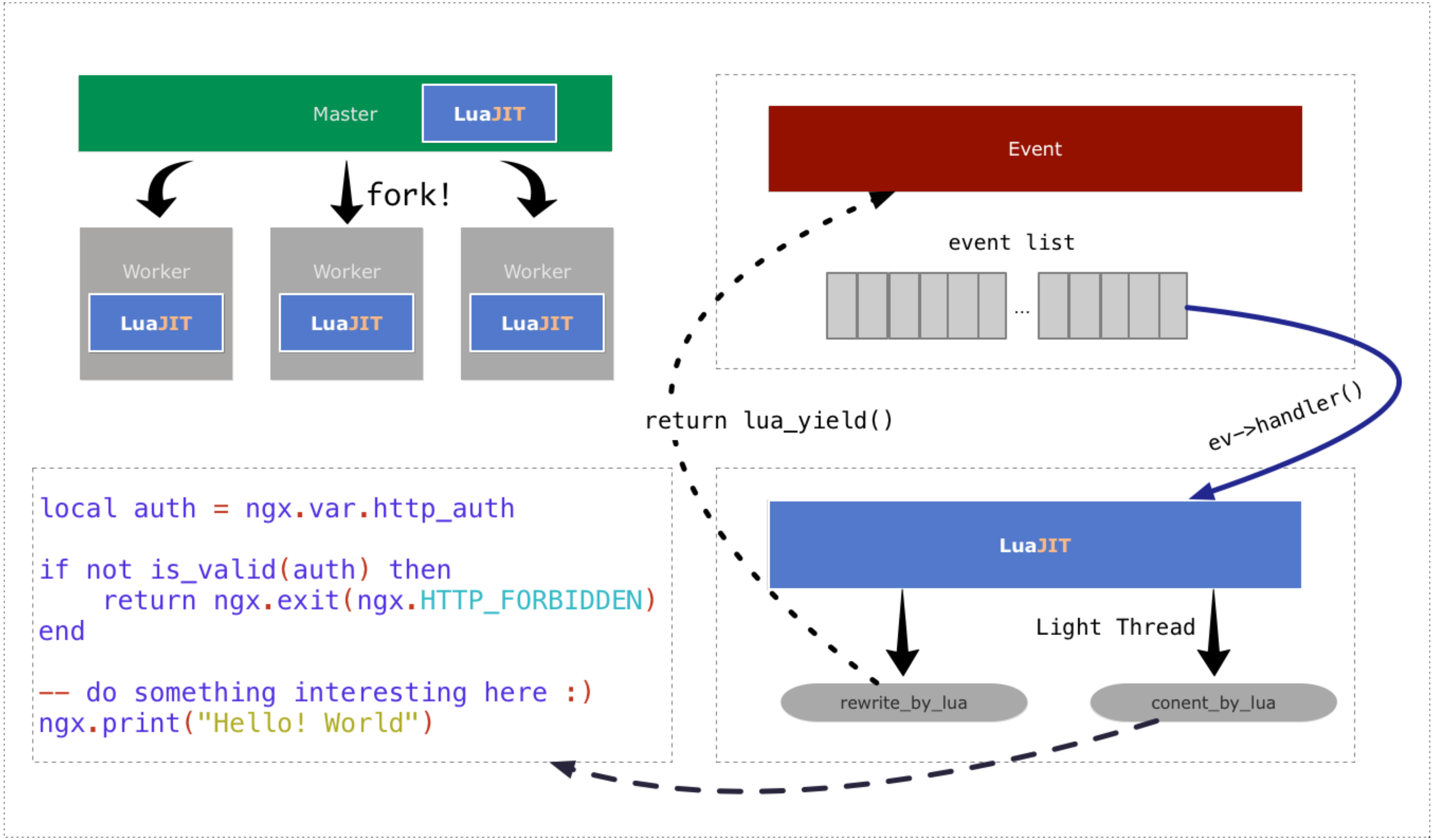
- **直观性**
- **交互性**
- https://github.com/openresty/openresty-systemtap-toolkit
- https://github.com/openresty/stapxx
- **On**-CPU & **Off**-CPU

Off-CPU

- http://www.brendangregg.com/offcpuanalysis.html



基于 OpenResty 的项目有何特点?



```lua
local auth = ngx.var.http_auth

if not is_valid(auth) then
    return ngx.exit(ngx.HTTP_FORBIDDEN)
end

-- do something interesting here :)
ngx.print("Hello! World")
```

- 多 **worker** 模式
- **Nginx** 事件循环 + 上层 **Lua VM** 接管
- 单线程，一个时刻只有**一个请求**在被处理
- 一个请求可能会经过**多次调度**之后才完成
- 分阶段的**流水线处理**（11 个阶段）
- 各阶段的 Lua code 运行在不同的 **Lua 协程**上

- 阻塞**事件循环**

- **锁抢占**

- **ngx.ctx** VS **ngx.var.VARIABLE**

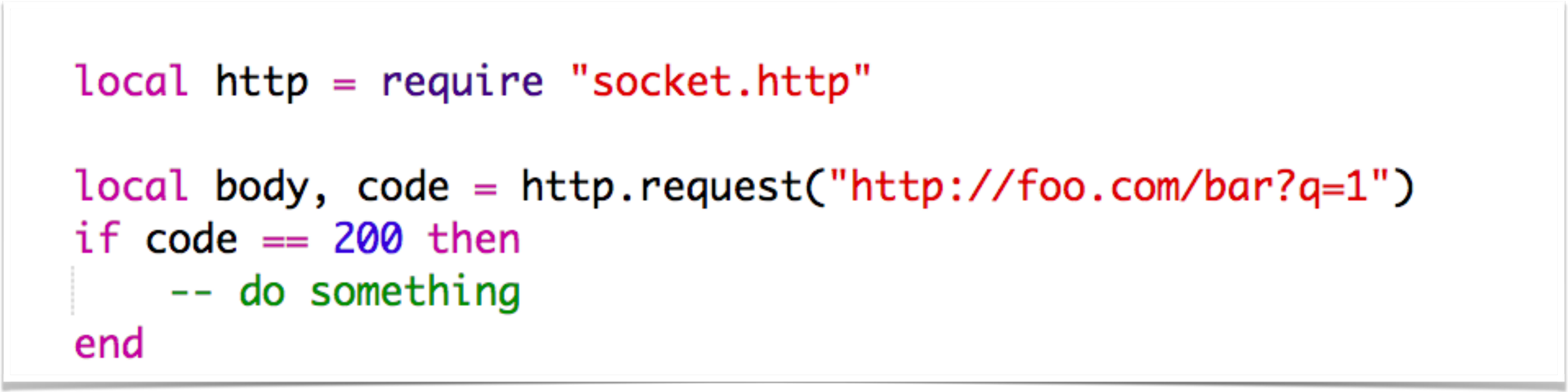
- 日志

- **LuaJIT** 的优势

- 编程习惯

阻塞事件循环

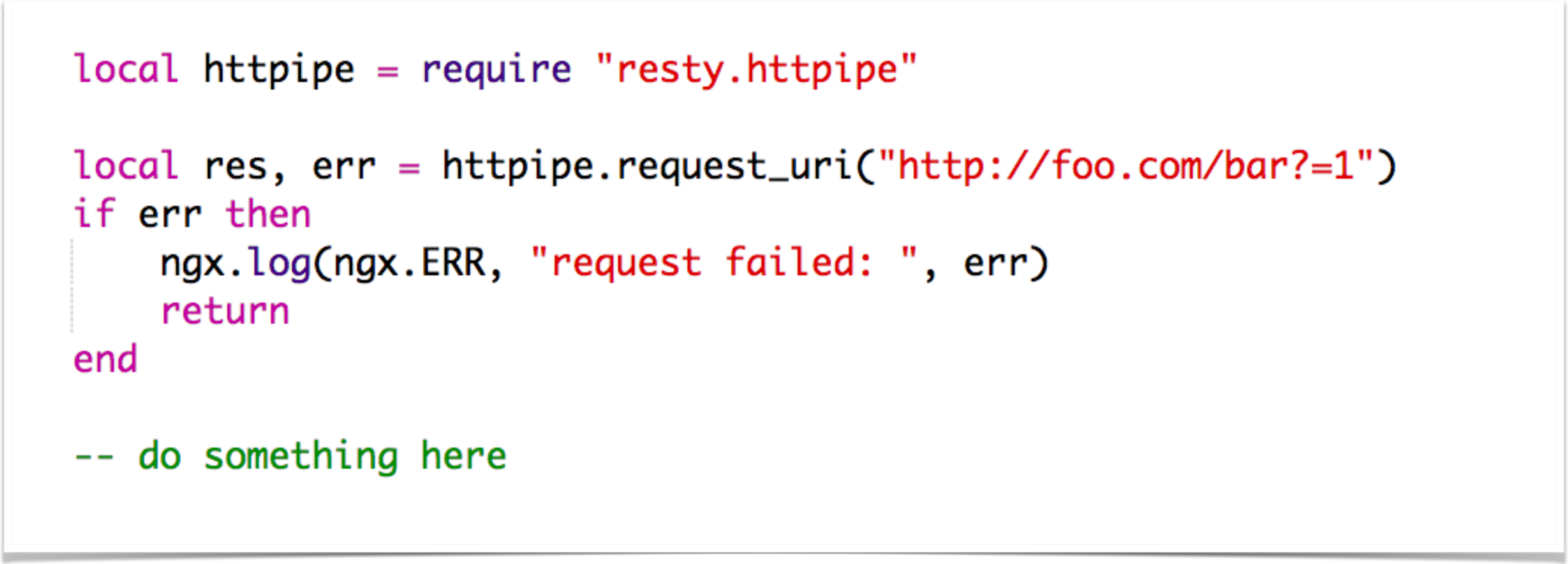
引用了一些 **Lua/C 第三方库**



```lua
local http = require "socket.http"

local body, code = http.request("http://foo.com/bar?q=1")
if code == 200 then
    -- do something
end
```

怎么解决?

使用 Cosocket

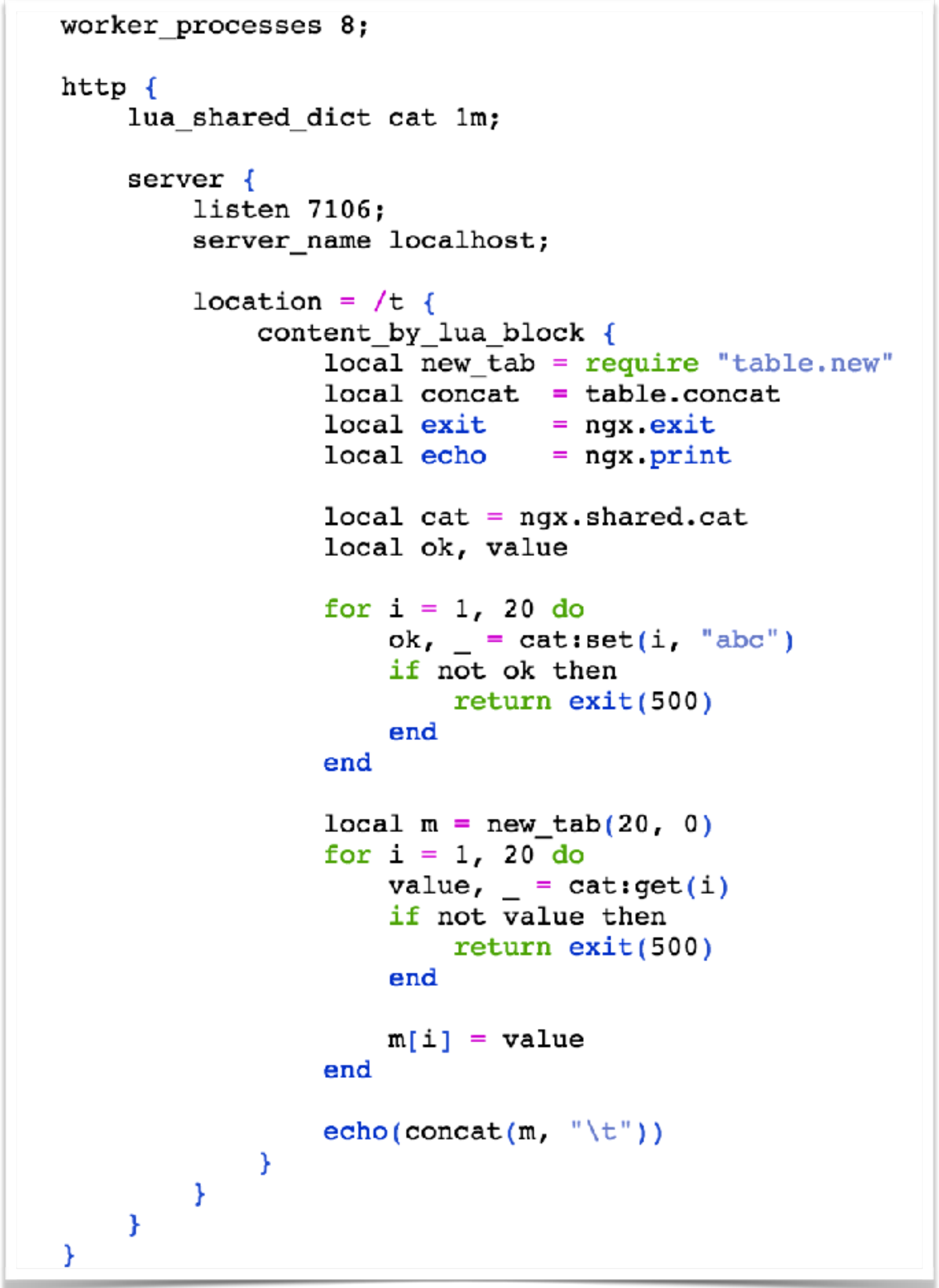


```lua
local httpipe = require "resty.httpipe"

local res, err = httpipe.request_uri("http://foo.com/bar?=1")
if err then
    ngx.log(ngx.ERR, "request failed: ", err)
    return
end

-- do something here
```

锁抢占

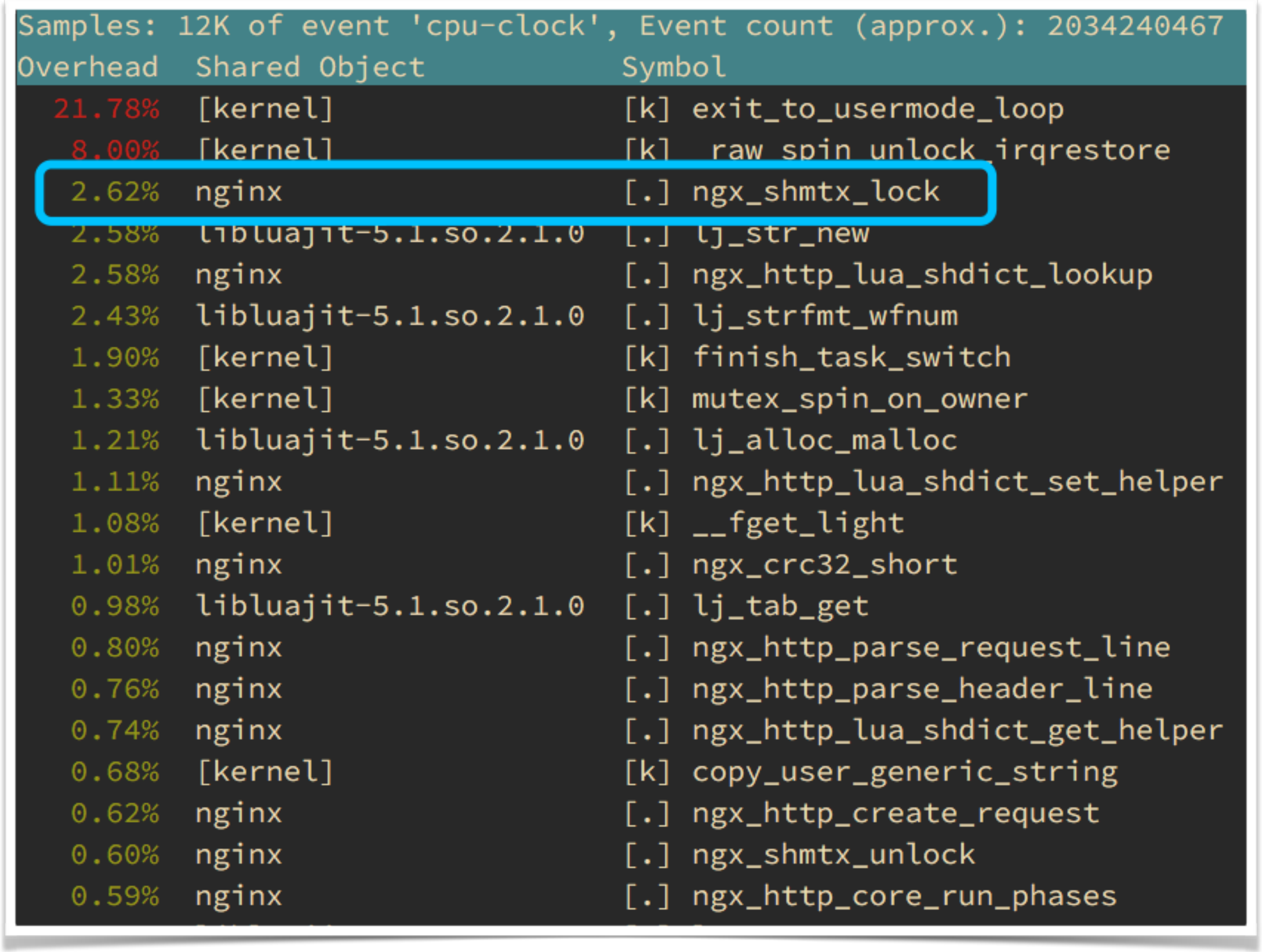
- ngx.shared.**DICT** — e.g ngx.shared.DICT.**get_keys**()
- nginx **cache**

Benchmark



```
worker_processes 8;

http {
    lua_shared_dict cat 1m;

    server {
        listen 7106;
        server_name localhost;

        location = /t {
            content_by_lua_block {
                local new_tab = require "table.new"
                local concat  = table.concat
                local exit    = ngx.exit
                local echo    = ngx.print

                local cat = ngx.shared.cat
                local ok, value

                for i = 1, 20 do
                    ok, _ = cat:set(i, "abc")
                    if not ok then
                        return exit(500)
                    end
                end

                local m = new_tab(20, 0)
                for i = 1, 20 do
                    value, _ = cat:get(i)
                    if not value then
                        return exit(500)
                    end

                    m[i] = value
                end

                echo(concat(m, "\t"))
            }
        }
    }
}
```

wrk -d 60s -t 4 -c 192 http://127.0.0.1:7106/t



```
Samples: 12K of event 'cpu-clock', Event count (approx.): 2034240467
Overhead  Shared Object          Symbol
  21.78%  [kernel]               [k] exit_to_usermode_loop
   8.00%  [kernel]               [k] _raw_spin_unlock_irqrestore
   2.62%  nginx                  [.] ngx_shmtx_lock
   2.58%  libluajit-5.1.so.2.1.0 [.] lj_str_new
   2.58%  nginx                  [.] ngx_http_lua_shdict_lookup
   2.43%  libluajit-5.1.so.2.1.0 [.] lj_strfmt_wfnum
   1.90%  [kernel]               [k] finish_task_switch
   1.33%  [kernel]               [k] mutex_spin_on_owner
   1.21%  libluajit-5.1.so.2.1.0 [.] lj_alloc_malloc
   1.11%  nginx                  [.] ngx_http_lua_shdict_set_helper
   1.08%  [kernel]               [k] __fget_light
   1.01%  nginx                  [.] ngx_crc32_short
   0.98%  libluajit-5.1.so.2.1.0 [.] lj_tab_get
   0.80%  nginx                  [.] ngx_http_parse_request_line
   0.76%  nginx                  [.] ngx_http_parse_header_line
   0.74%  nginx                  [.] ngx_http_lua_shdict_get_helper
   0.68%  [kernel]               [k] copy_user_generic_string
   0.62%  nginx                  [.] ngx_http_create_request
   0.60%  nginx                  [.] ngx_shmtx_unlock
   0.59%  nginx                  [.] ngx_http_core_run_phases
```

0x681..
0x4deff..
0x4e0e..
0x6812..
0x4e6..
0x5357..
0x4e9..
0x4e0af..
0x4eb..
0x4e1136 ..
0x5e37..
0x4e72d4 ..
0x57fdf..
0x57fe74 ..
0x581..
0x594f2e ..
0x4ec9..
0x456b71 ..
0x4ee3..
0x4ec3ce ..
0x4a23..
0x4de08a ..
0x576e..
0x4eeed0 ..
0x576f..
0x4a23c0 ..
0x5624..
0x4b169b ..
0x561c..
0x577202 ..
0x7f16b..
0x5770f2 ..
0x577dd..
0x578a46 ..
0x57da0f : ngx_http_..
0x57deae : ngx_http_..
0x57dc64 : ngx_http_..
0x4a1084 : ngx_http_..
0x49ff9b : ngx_http_c..
0x49ff09 : ngx_http_h..
0x4aee55 : ngx_http_..
0x4ad92b : ngx_http_..
0x4aced5 : ngx_http_..
0x4b1296 : ngx_http_..
0x48f99d : ngx_epoll_p..
0x480da7 : ngx_proces..
0x48d73e : ngx_worker..
0x48a5b6 : ngx_spawn..
0x48c961 : ngx_start_..
0..
0x48c4e6 : ngx_master..
0x7f16b9d00a00 : sem_wait
0x7f16b9d00a00 :..
0x7f16..
0..
0x44f869 : main
0x468b4f : ngx_shmtx_lock
0x468b4f : ngx_s..
0x468b..
0x7f16b9..
0..
0x7f16b924ad1d : __lib..
0x585b38 : ngx_http_lua_ffi_shdict_store
0x585b38 : ngx_h..
0x586..
0x468b4f ..
0..
0x44f141 : _start
0x46..
0x468a4..
0x4..
0x7f16ba47e79f : 0x7f16ba47e79f
0x7f16ba47e8cd :..
0x7f16..
0x7f16ba..
all

Function: 0x468b4f : ngx_shmtx_lock (258,264 samples, 27.16%)

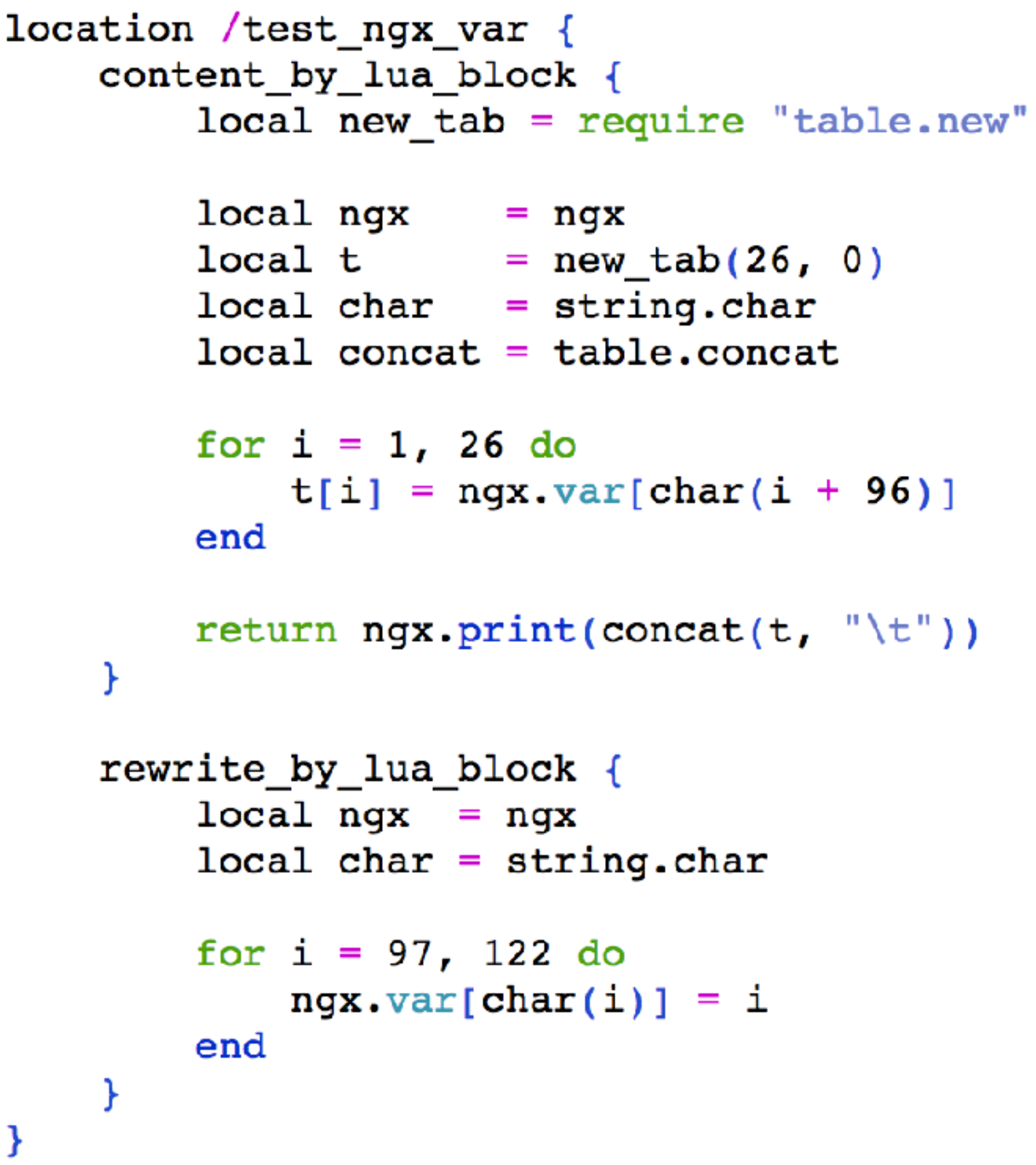
ngx.ctx VS ngx.var.VARIABLE

- **ngx.ctx** 是一个“神奇”的 **Lua table**，而用法和普通 Lua table 一致
- **ngx.var.VARIABLE** 利用了 nginx 的**变量系统**，同样可以用于存储信息
- **ngx.ctx** 拥有比 **ngx.var.VARIABLE** 更好的效率

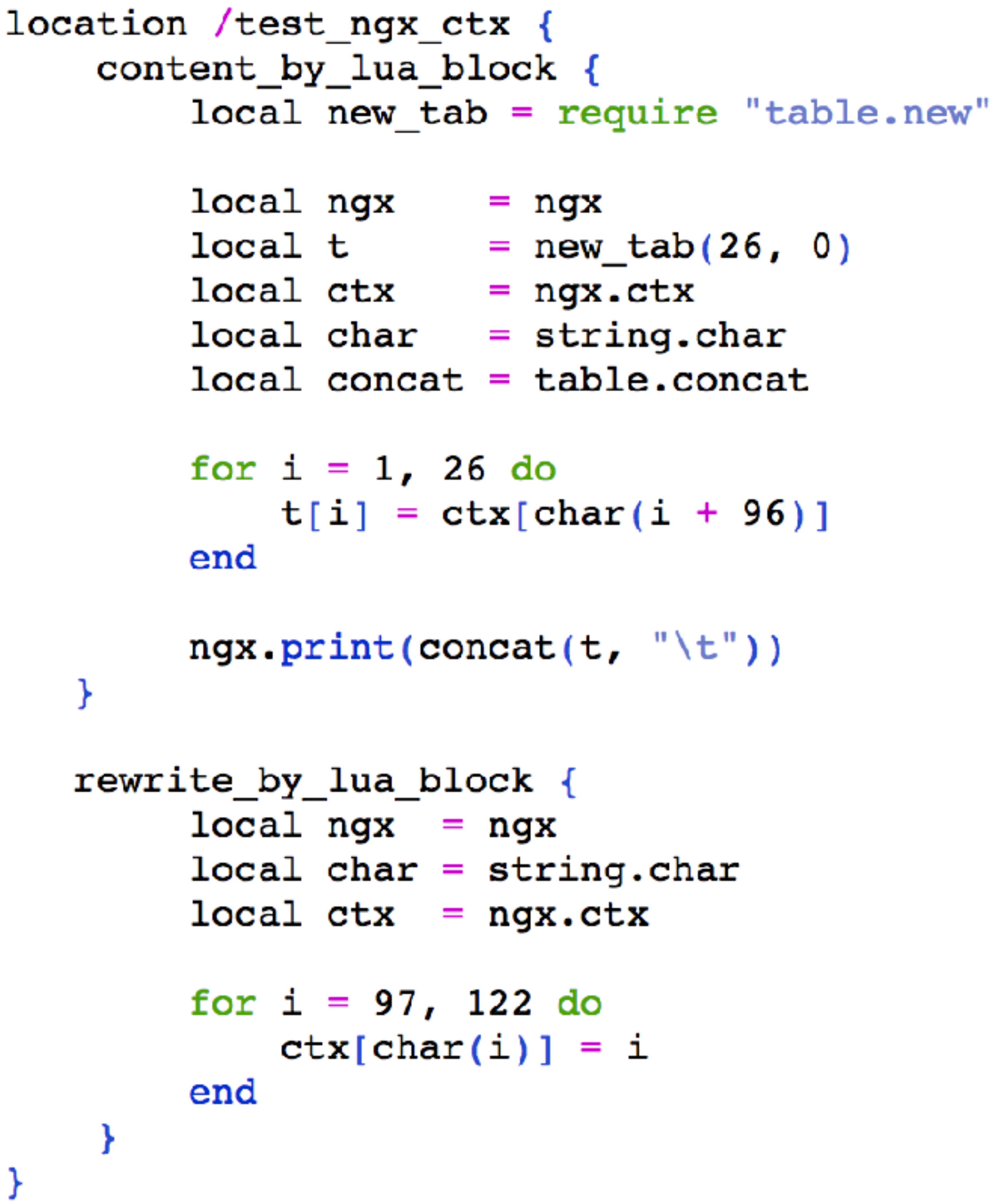
Why **ngx.ctx** is better

- nginx 变量只有**字符串**一种类型
- nginx 变量需要**分配内存**用于存放变量值信息，且**只能在请求结束时被释放**
- **Lua table** 具有非常高的查找效率

Benchmark



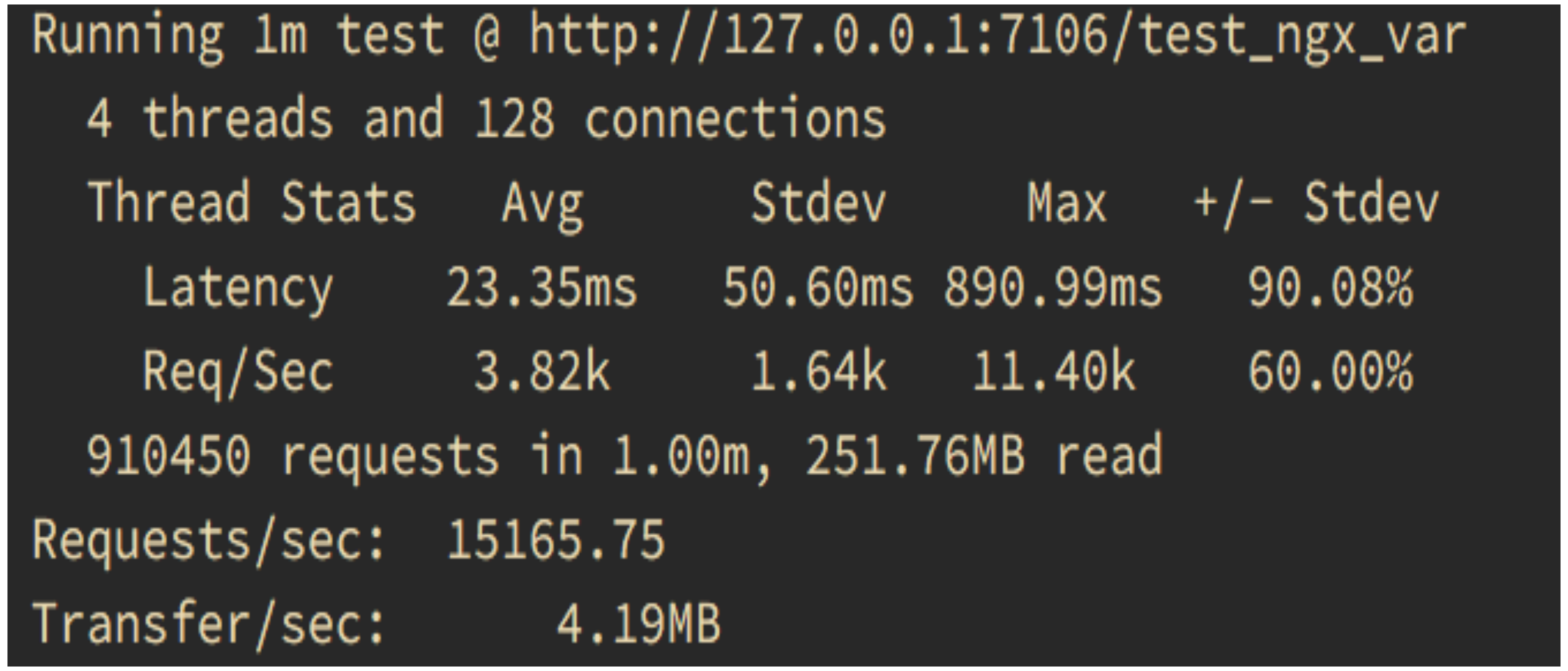
```
location /test_ngx_var {
    content_by_lua_block {
        local new_tab = require "table.new"

        local ngx    = ngx
        local t      = new_tab(26, 0)
        local char   = string.char
        local concat = table.concat

        for i = 1, 26 do
            t[i] = ngx.var[char(i + 96)]
        end

        return ngx.print(concat(t, "\t"))
    }

    rewrite_by_lua_block {
        local ngx  = ngx
        local char = string.char

        for i = 97, 122 do
            ngx.var[char(i)] = i
        end
    }
}
```



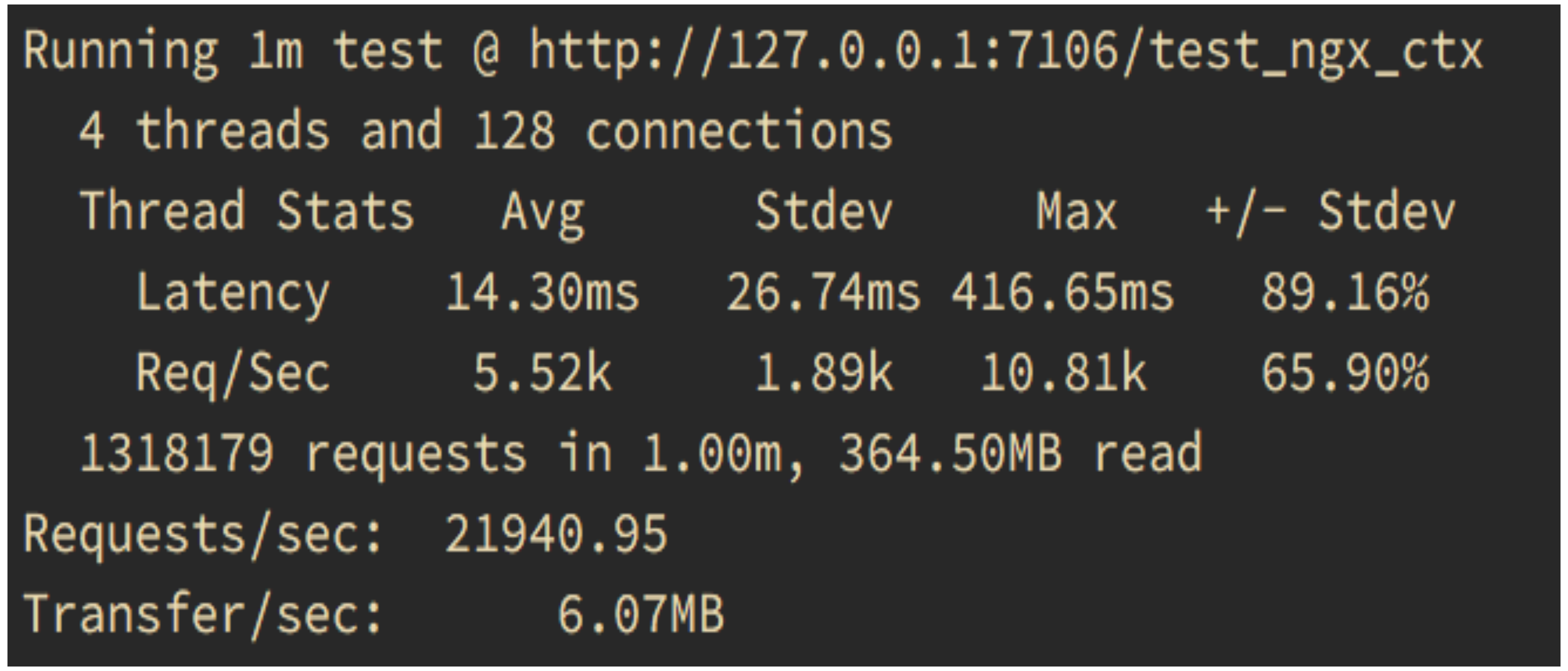
```
location /test_ngx_ctx {
    content_by_lua_block {
        local new_tab = require "table.new"

        local ngx    = ngx
        local t      = new_tab(26, 0)
        local ctx    = ngx.ctx
        local char   = string.char
        local concat = table.concat

        for i = 1, 26 do
            t[i] = ctx[char(i + 96)]
        end

        ngx.print(concat(t, "\t"))
   }

   rewrite_by_lua_block {
        local ngx  = ngx
        local char = string.char
        local ctx  = ngx.ctx

        for i = 97, 122 do
            ctx[char(i)] = i
        end
    }
}
```

wrk -d 60s -t 4 -c 128 http://127.0.0.1:7106/test_ngx_var

wrk -d 60s -t 4 -c 128 http://127.0.0.1:7106/test_ngx_ctx



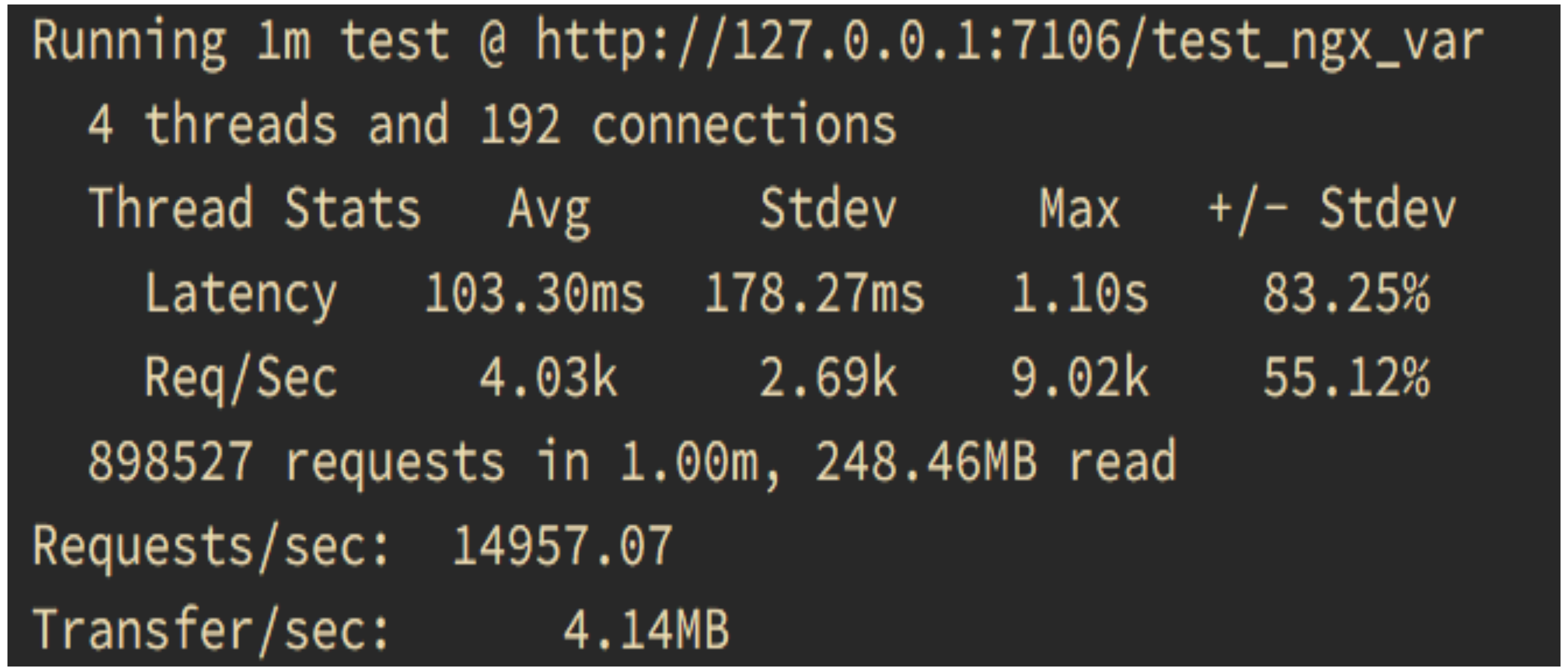
```
Running 1m test @ http://127.0.0.1:7106/test_ngx_var
  4 threads and 128 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    23.35ms   50.60ms 890.99ms   90.08%
    Req/Sec     3.82k     1.64k   11.40k    60.00%
  910450 requests in 1.00m, 251.76MB read
Requests/sec:  15165.75
Transfer/sec:      4.19MB
```



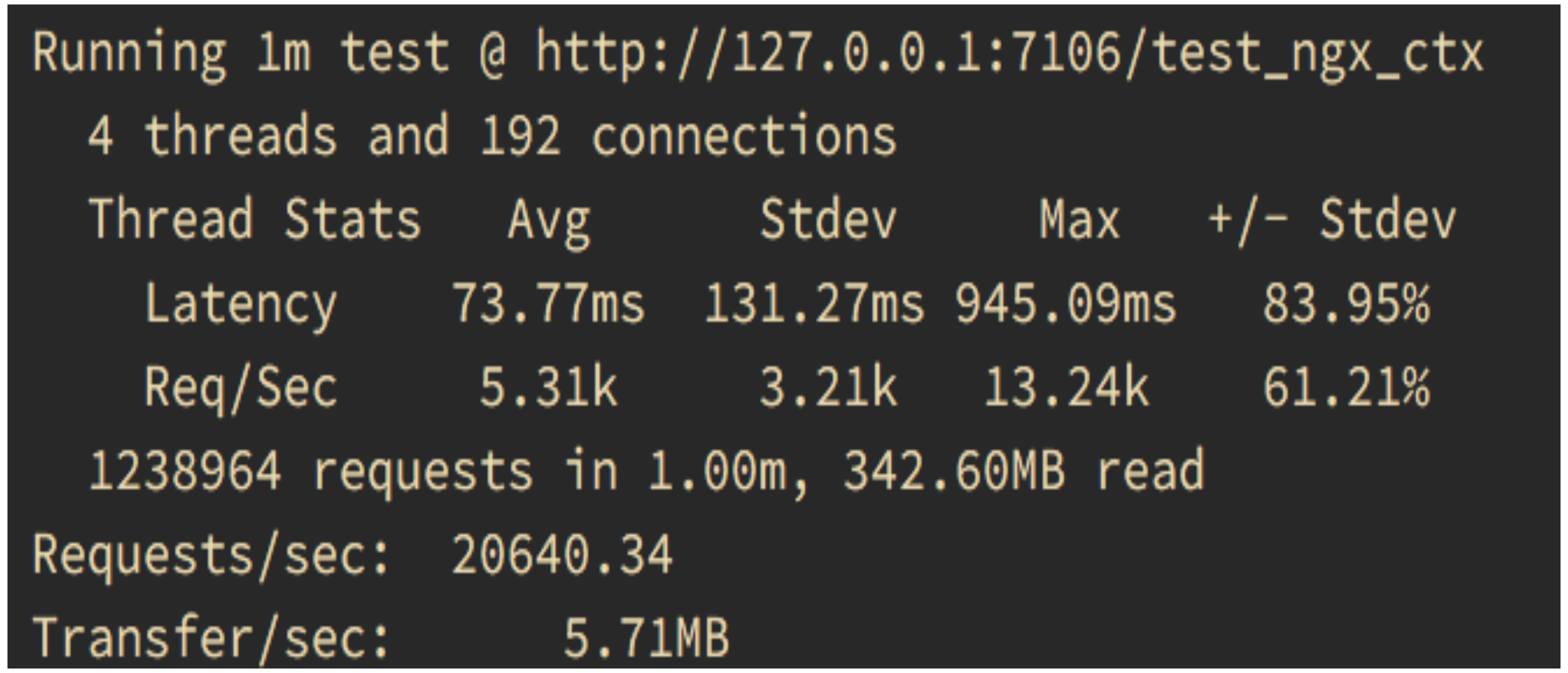
```
Running 1m test @ http://127.0.0.1:7106/test_ngx_ctx
  4 threads and 128 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    14.30ms   26.74ms 416.65ms   89.16%
    Req/Sec     5.52k     1.89k   10.81k    65.90%
  1318179 requests in 1.00m, 364.50MB read
Requests/sec:  21940.95
Transfer/sec:      6.07MB
```

wrk -d 60s -t 4 -c 192 http://127.0.0.1:7106/test_ngx_var

wrk -d 60s -t 4 -c 192 http://127.0.0.1:7106/test_ngx_ctx



```
Running 1m test @ http://127.0.0.1:7106/test_ngx_var
  4 threads and 192 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   103.30ms  178.27ms   1.10s    83.25%
    Req/Sec     4.03k     2.69k    9.02k    55.12%
  898527 requests in 1.00m, 248.46MB read
Requests/sec:  14957.07
Transfer/sec:      4.14MB
```

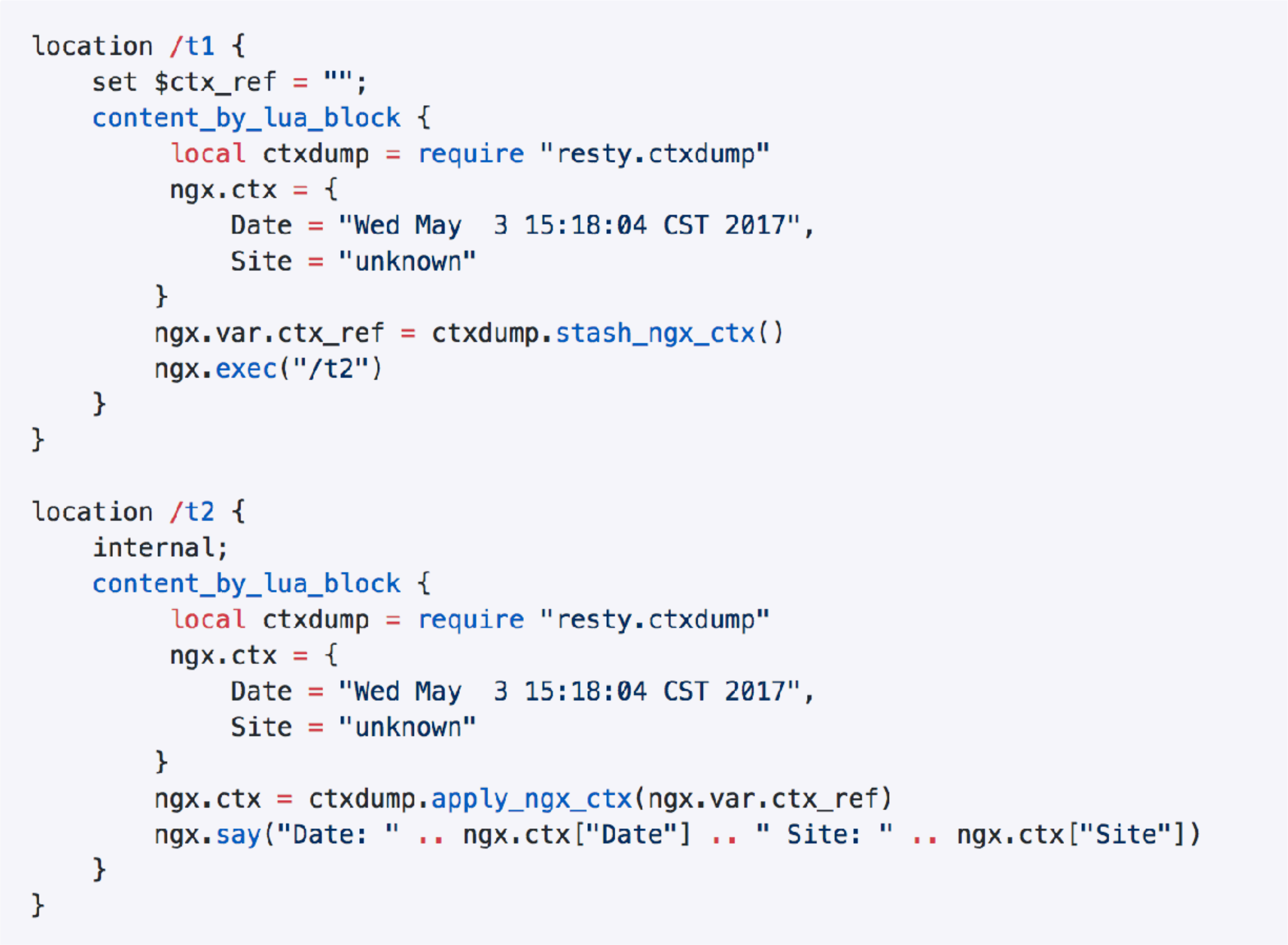


```
Running 1m test @ http://127.0.0.1:7106/test_ngx_ctx
  4 threads and 192 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    73.77ms  131.27ms 945.09ms   83.95%
    Req/Sec     5.31k     3.21k   13.24k    61.21%
  1238964 requests in 1.00m, 342.60MB read
Requests/sec:  20640.34
Transfer/sec:      5.71MB
```

ngx.ctx 的不足

- 相对昂贵的 **metamethod** 调用 - 集中使用时局部缓存
- 生命周期局限在一个 **location** - https://github.com/tokers/lua-resty-ctxdump

lua-resty-ctxump



```
location /t1 {
    set $ctx_ref = "";
    content_by_lua_block {
         local ctxdump = require "resty.ctxdump"
         ngx.ctx = {
             Date = "Wed May  3 15:18:04 CST 2017",
             Site = "unknown"
        }
        ngx.var.ctx_ref = ctxdump.stash_ngx_ctx()
        ngx.exec("/t2")
    }
}

location /t2 {
    internal;
    content_by_lua_block {
         local ctxdump = require "resty.ctxdump"
         ngx.ctx = {
             Date = "Wed May  3 15:18:04 CST 2017",
             Site = "unknown"
        }
        ngx.ctx = ctxdump.apply_ngx_ctx(ngx.var.ctx_ref)
        ngx.say("Date: " .. ngx.ctx["Date"] .. " Site: " .. ngx.ctx["Site"])
    }
}
```

日志

- 合理设置 **access_log** 的 buffer 大小 - 避免过多的 write 系统调用
- 关闭 **access_log 和拦截 error_log，经过网络传输到外部组件**

利用 LuaJIT 的优势

- 引入 **lua-resty-core**（https://github.com/openresty/lua-resty-core）
- 使用可被 **JIT** 编译器编译的函数（http://wiki.luajit.org/NYI）
- 尽量避免 **table resize**（table.new）

良好的编程习惯

- https://blog.codingnow.com/cloud/LuaTips
- 避免滥用**全局变量**
- 避免低效率的字符串拼接 - **table.concat**

upyun-resty

又拍云

- https://github.com/upyun/upyun-resty
- **Tech** Talks
- **Nginx** Modules
- **Lua-Resty** Libraries
- **Projects**

Thanks