Optimizing Lua Applications for LuaJIT and OpenResty

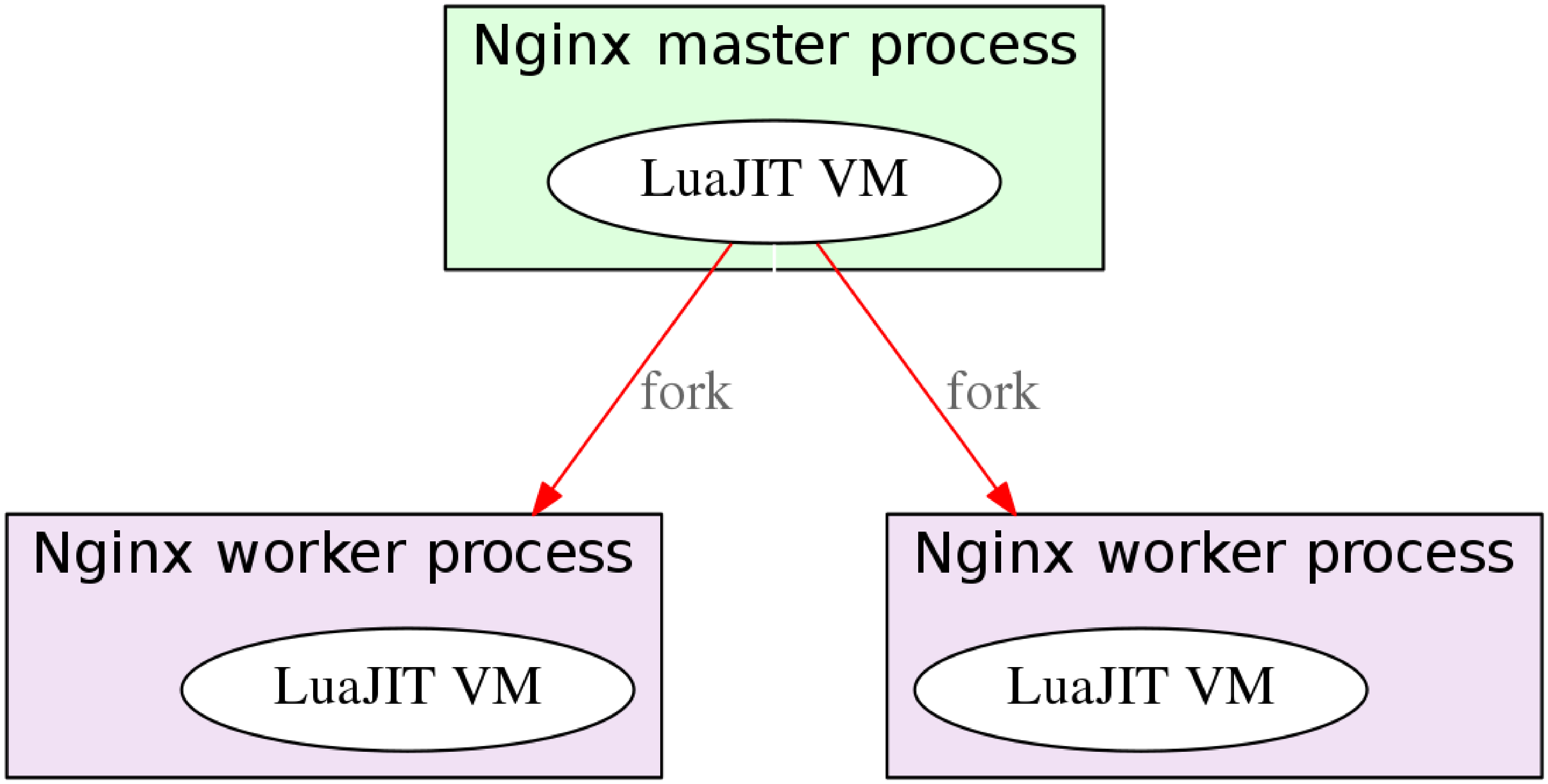
☺ *agentzh@openresty.org* ☺

Yichun Zhang (@agentzh)

2016.9

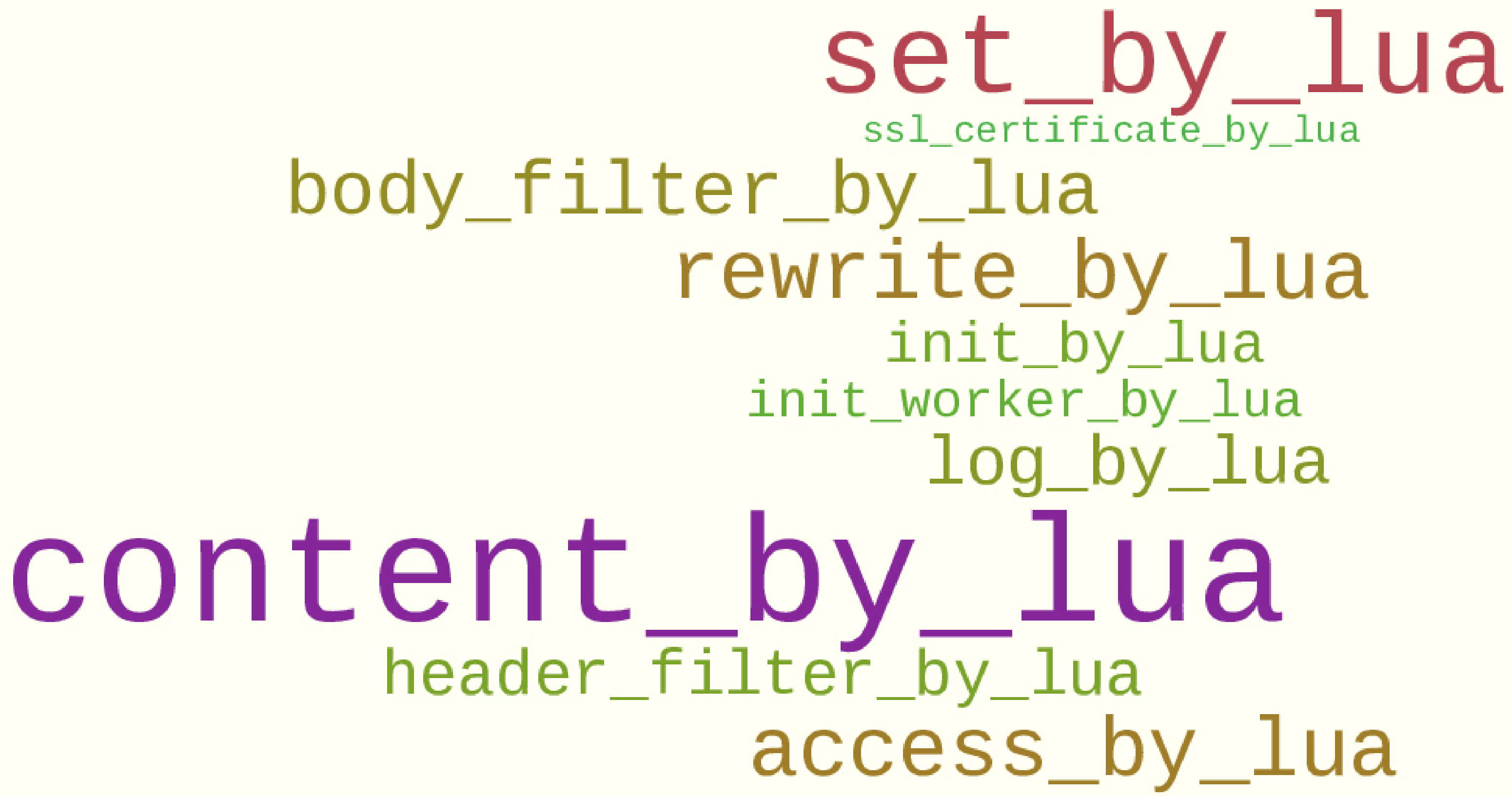
OPENRESTY

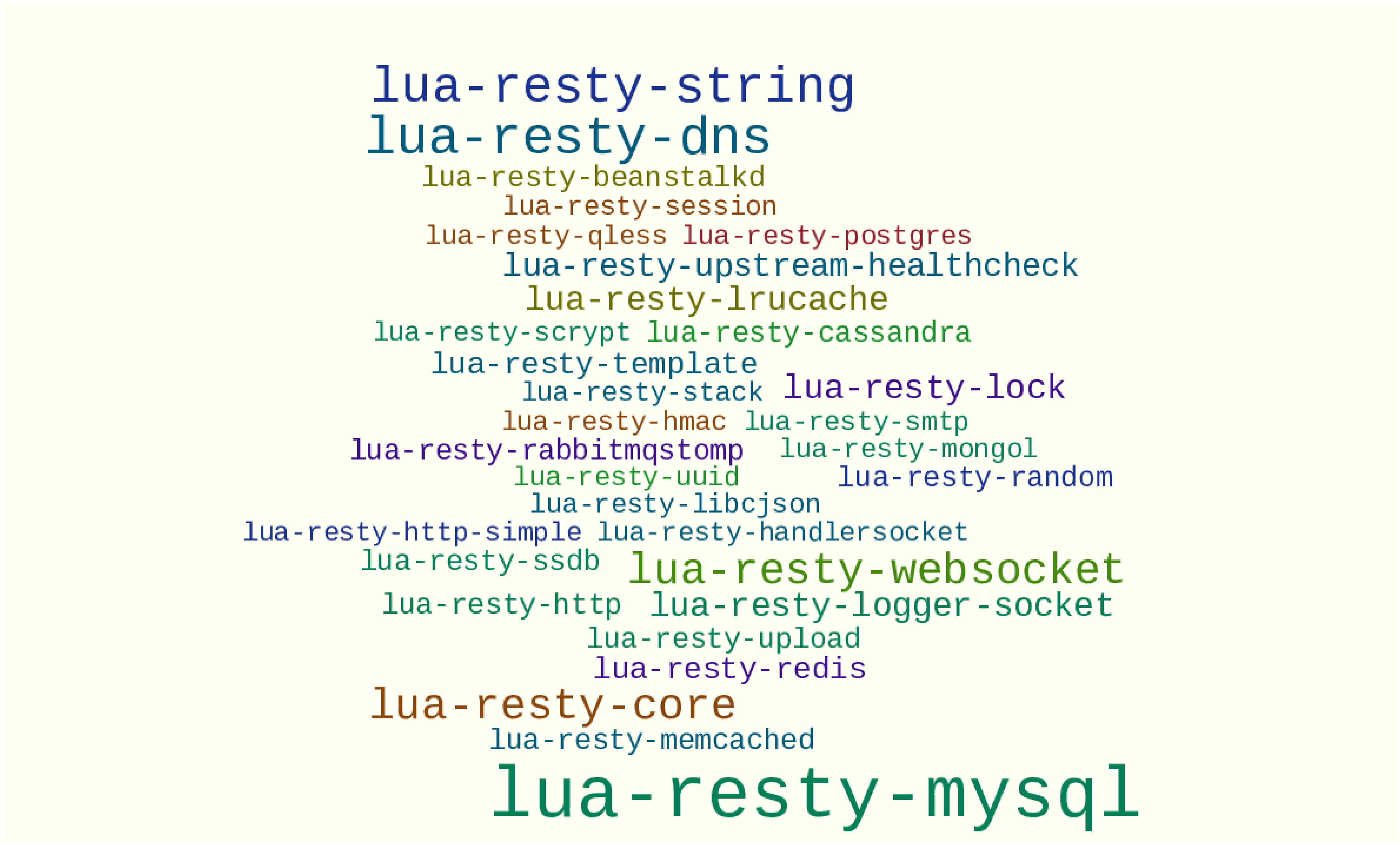
♡ NGINX + LuaJIT



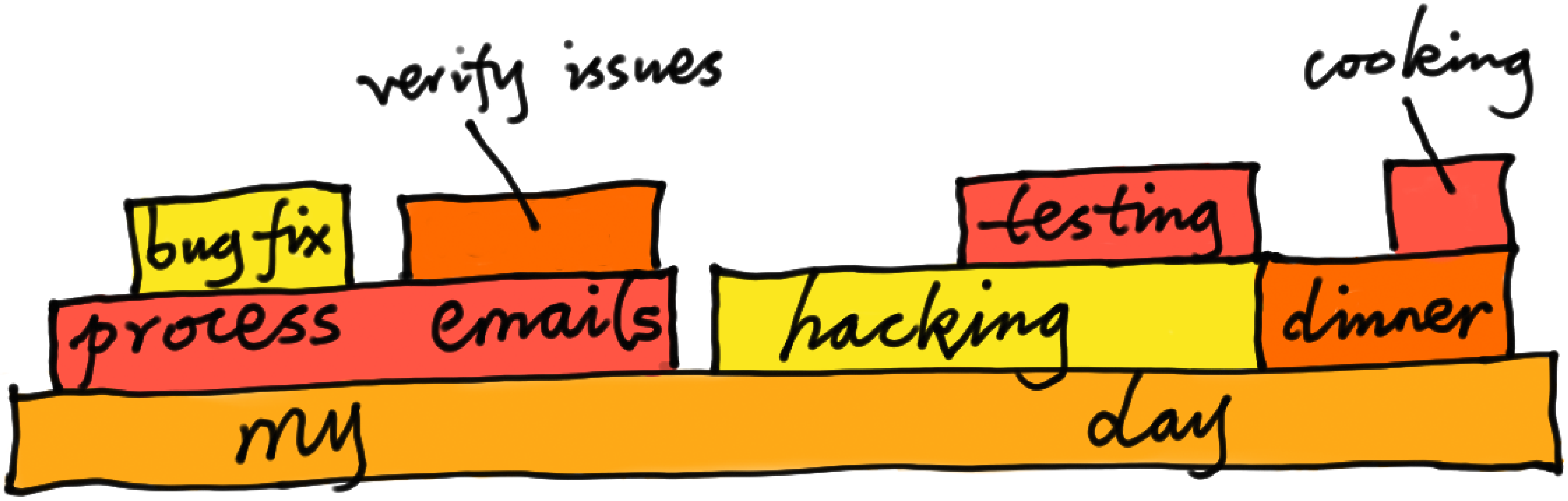
How ngx_lua works

Click anywhere or press a button to close

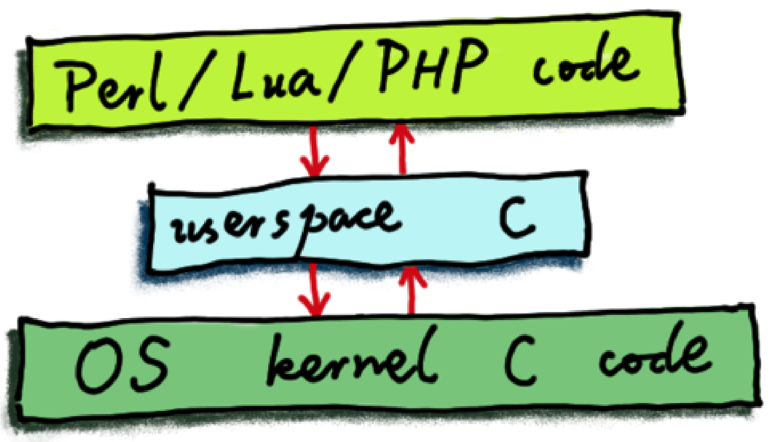



lua-resty-string
lua-resty-dns
lua-resty-beanstalkd
lua-resty-session
lua-resty-qless
lua-resty-postgres
lua-resty-upstream-healthcheck
lua-resty-lrucache
lua-resty-scrypt
lua-resty-cassandra
lua-resty-template
lua-resty-stack
lua-resty-lock
lua-resty-hmac
lua-resty-smtp
lua-resty-rabbitmqstomp
lua-resty-mongol
lua-resty-uuid
lua-resty-random
lua-resty-libcjson
lua-resty-http-simple
lua-resty-handlersocket
lua-resty-ssdb
lua-resty-websocket
lua-resty-http
lua-resty-logger-socket
lua-resty-upload
lua-resty-redis
lua-resty-core
lua-resty-memcached
lua-resty-mysql

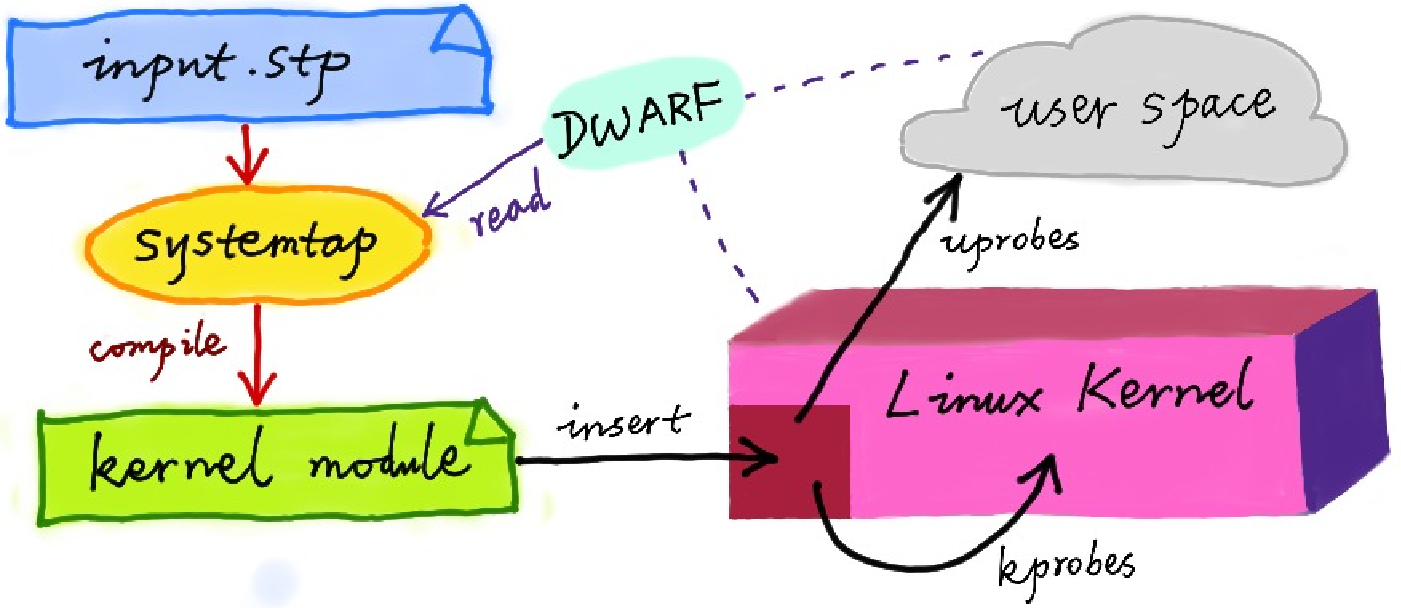
☺ Flame Graphs



Flame Graph for My Day

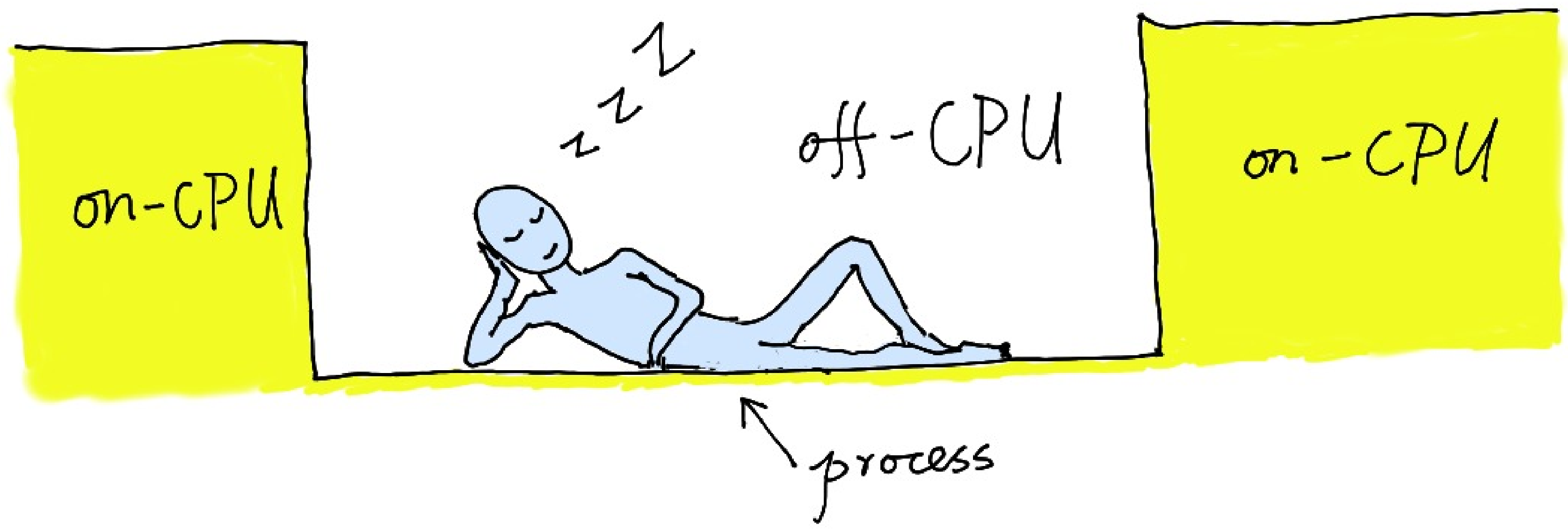


The Software Stack


input.stp
DWARF
user space
read
systemtap
uprobes
compile
Linux Kernel
insert
kernel module
kprobes

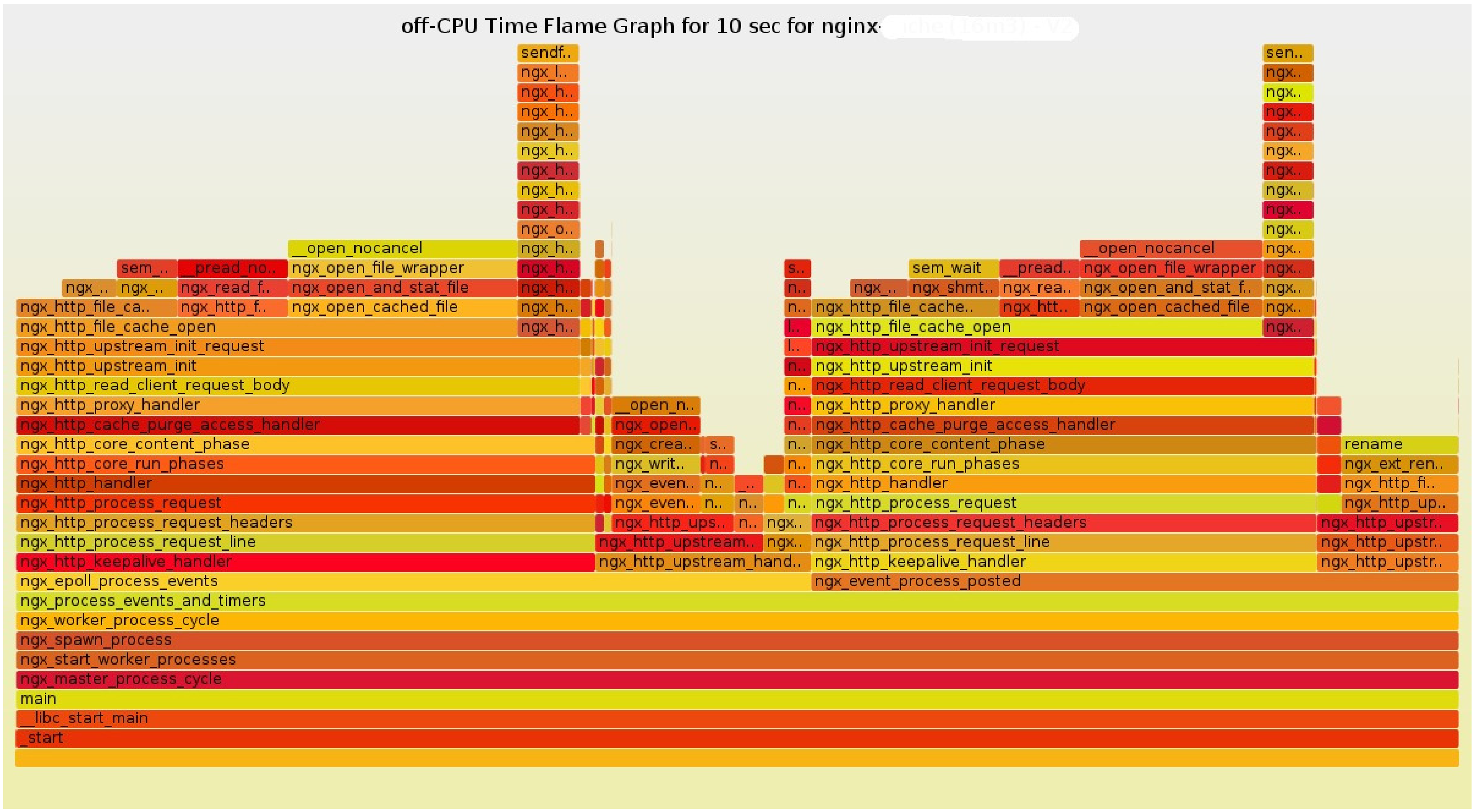
☺ I/O

♡ *Off*-CPU Flame Graphs


on-CPU
z z Z
off-CPU
on-CPU
process

```
# assuming the nginx worker process to be analyzed is 10901.
./sample-bt-off-cpu -p 10901 -t 5 > a.bt
```

```
# using Brendan Gregg's flame graph tools:
$ stackcollapse-stap.pl a.bt > a.cbt
$ flamegraph.pl a.cbt > a.svg
```

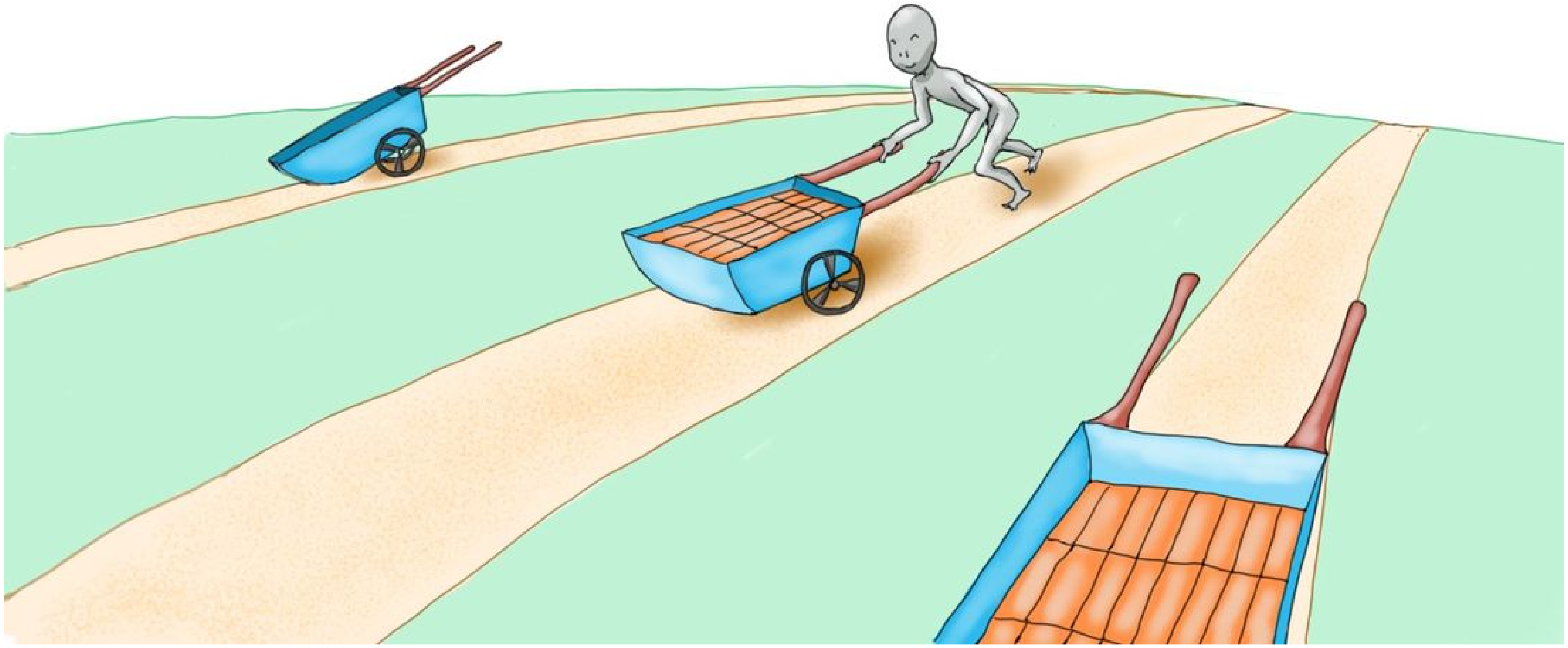

off-CPU Time Flame Graph for 10 sec for nginx-[redacted]
sendf..
sen..
ngx_l..
ngx_h..
ngx_o..
ngx..
__open_nocancel
sem_..
__pread_no..
ngx_open_file_wrapper
ngx_..
ngx_read_f..
ngx_open_and_stat_file
ngx_http_file_ca..
ngx_http_f..
ngx_open_cached_file
ngx_http_file_cache_open
ngx_http_upstream_init_request
ngx_http_upstream_init
ngx_http_read_client_request_body
ngx_http_proxy_handler
ngx_http_cache_purge_access_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
__open_n..
ngx_open..
ngx_crea..
s..
ngx_writ..
n..
ngx_even..
ngx_http_ups..
ngx_http_upstream..
ngx_http_upstream_hand..
ngx..
l..
sem_wait
__pread..
ngx_shmt..
ngx_rea..
ngx_open_and_stat_f..
ngx_http_file_cache..
ngx_htt..
rename
ngx_ext_ren..
ngx_http_fi..
ngx_http_up..
ngx_http_upstr..
ngx_epoll_process_events
ngx_event_process_posted
ngx_process_events_and_timers
ngx_worker_process_cycle
ngx_spawn_process
ngx_start_worker_processes
ngx_master_process_cycle
main
__libc_start_main
_start

♡ Synchronously *nonblocking* I/O

♡ Light *threads* & semaphores

```
local thread_A, err =
        ngx.thread.spawn(func1)

-- thread_A keeps running asynchronously
-- in the background of the current
-- "light thread".
```

```lua
local ok, res1, res2 =
            ngx.thread.wait(thread_A, thread_B)
```



```lua
local ok, err = ngx.thread.kill(thread_A)
```

♡ Full-Duplex Cosockets

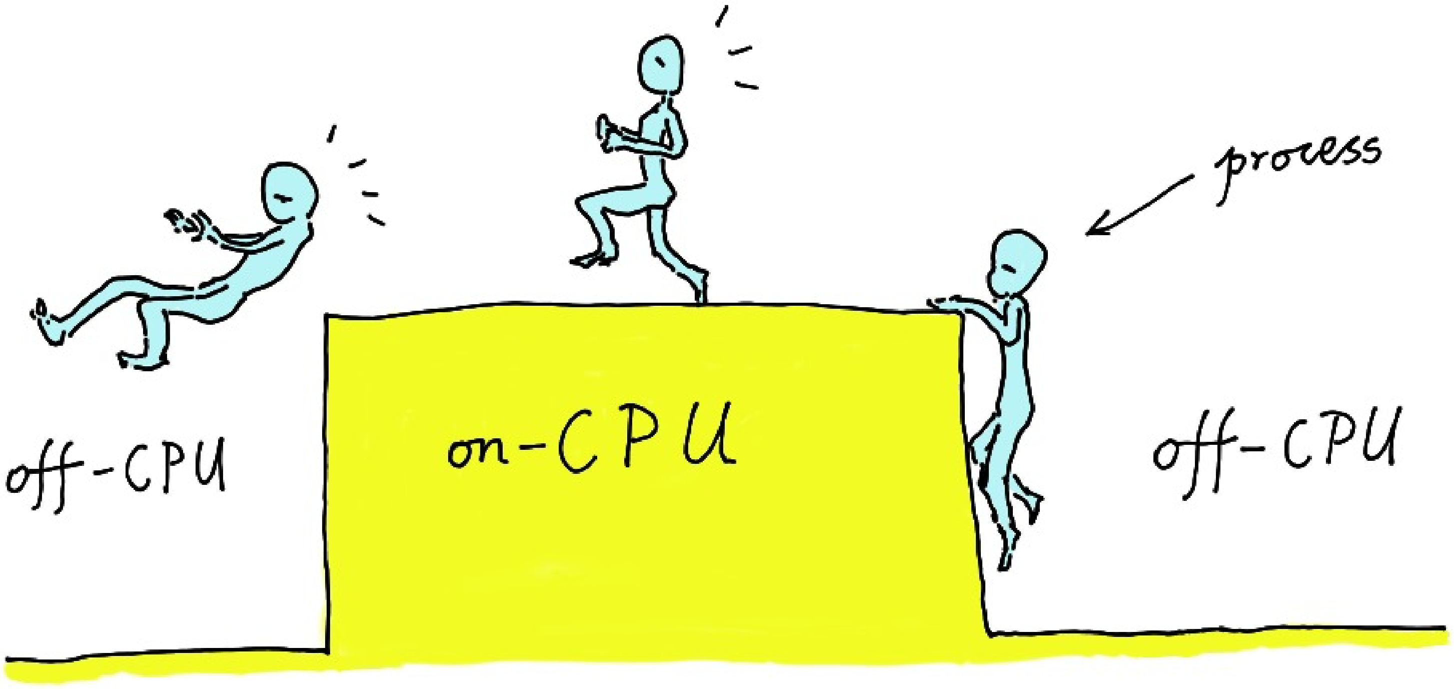
```
local sock = ngx.socket.tcp()
local ok, err = sock:connect("www.cloudflare.com",
                             443)

ok, err = sock:sslhandshake(
    false,  -- disable SSL session
    "www.cloudflare.com",  -- SNI name
    true  -- verify everything
)
```

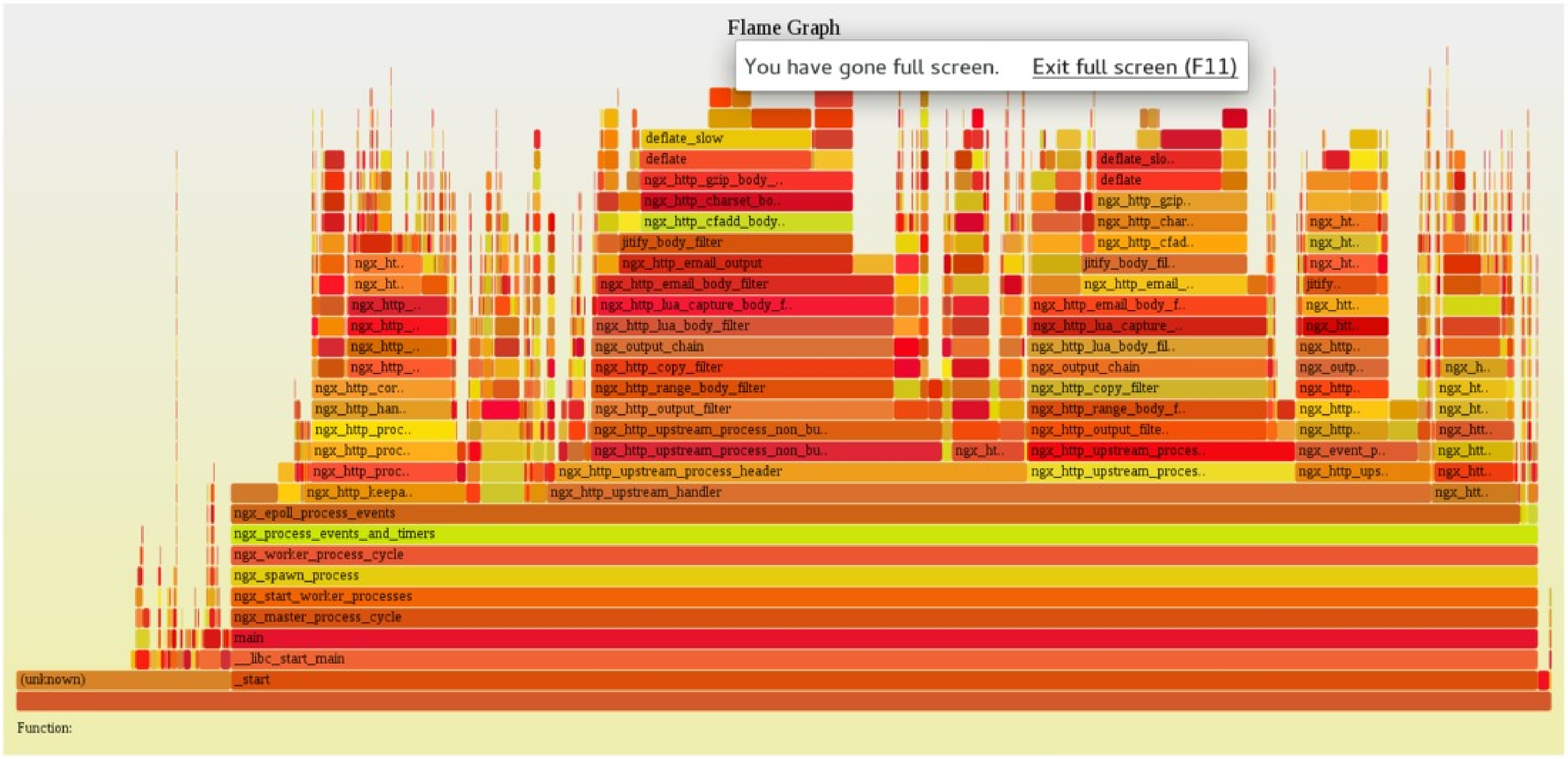
♡ Timers and Sleeps

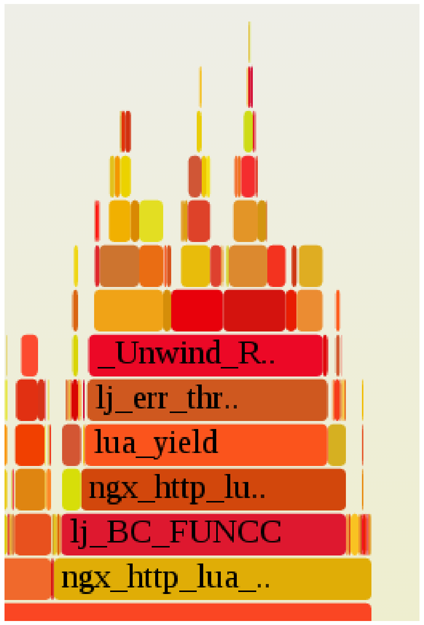
```
-- create a timer triggered after 1 sec
ngx.timer.at(1000, function (premature)
                             do_something()
                         end)

-- sleeps for 1 sec then continue
ngx.sleep(1000)
```

☺ CPU

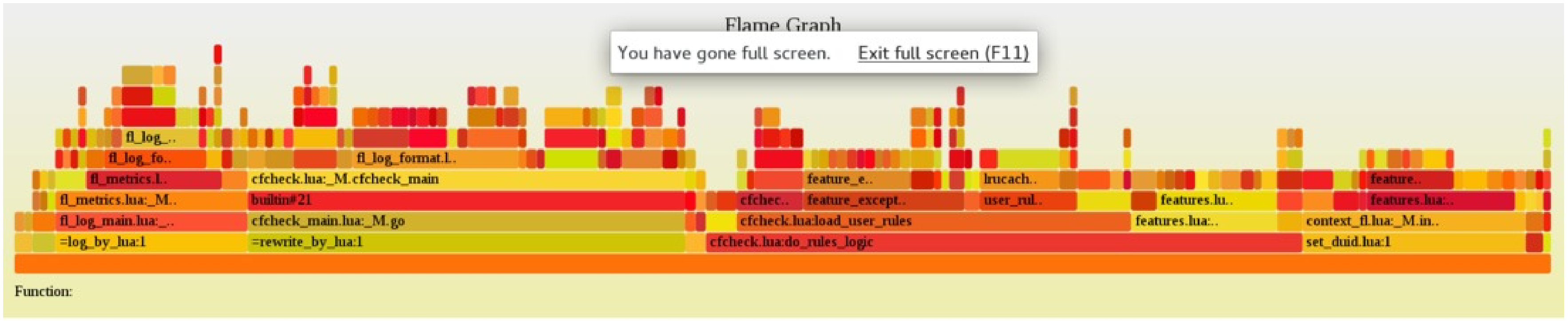
♡ *on*-CPU Flame Graphs


off-CPU
on-CPU
process
off-CPU


Flame Graph
You have gone full screen.
Exit full screen (F11)
deflate_slow
deflate
ngx_http_gzip_body_..
ngx_http_charset_bo..
ngx_http_cfadd_body..
jitify_body_filter
ngx_http_email_output
ngx_http_email_body_filter
ngx_http_lua_capture_body_f..
ngx_http_lua_body_filter
ngx_output_chain
ngx_http_copy_filter
ngx_http_range_body_filter
ngx_http_output_filter
ngx_http_upstream_process_non_bu..
ngx_http_upstream_process_non_bu..
ngx_http_upstream_process_header
ngx_http_upstream_handler
ngx_http_keepa..
ngx_http_proc..
ngx_http_han..
ngx_http_cor..
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
(unknown)
_start
Function:


_Unwind_R..
lj_err_thr..
lua_yield
ngx_http_lu..
lj_BC_FUNCC
ngx_http_lua_..

♡ Lua-land Flame Graphs


Flame Graph
You have gone full screen.
Exit full screen (F11)
fl_log_..
fl_log_fo..
fl_log_format.l..
fl_metrics.l..
cfcheck.lua:_M.cfcheck_main
feature_e..
lrucach..
feature..
fl_metrics.lua:_M..
builtin#21
cfchec..
feature_except..
user_rul..
features.lu..
features.lua:..
fl_log_main.lua:_..
cfcheck_main.lua:_M.go
cfcheck.lua:load_user_rules
features.lua:..
context_fl.lua:_M.in..
=log_by_lua:1
=rewrite_by_lua:1
cfcheck.lua:do_rules_logic
set_duid.lua:1
Function:

http://agentzh.org/misc/flamegraph/lua-on-cpu-local-waf-jitted-only.svg

```
lj-lua-stacks.sxx --arg time=5 \
                  --skip-badvars \
                  -x 6949 \
                  > a.bt
```

♡ LuaJIT Built-in Profiler

vs

SystemTap Sampling

♡ Dynamic Allocations & Garbage Collection

Lua tables

```
lj_tab_new

lj_tab_resize

lj_tab_len
```

```
table.new(10, 20)
table.clear(tb)
```

```
tb[key1] = val1
tb[key1] = nil
tb[key2] = val2
```

Lua strings

```
?    s = s .. r
```

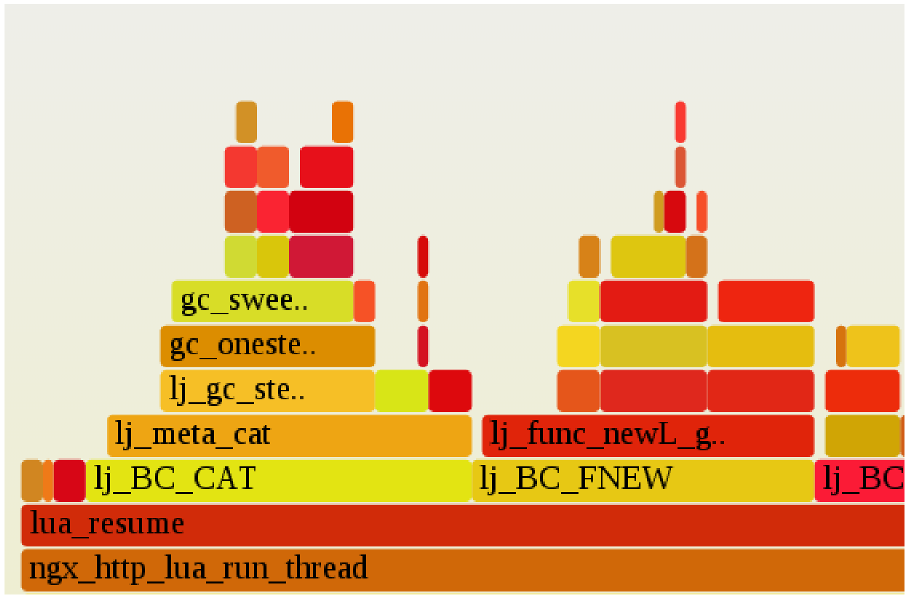
```lua
-- tb[#tb + 1] is slow!
idx = idx + 1
tb[idx] = r

s = table.concat(tb)
```

```
?    string.sub(s, i, i)
```

```
string.byte(s, i, i)
```

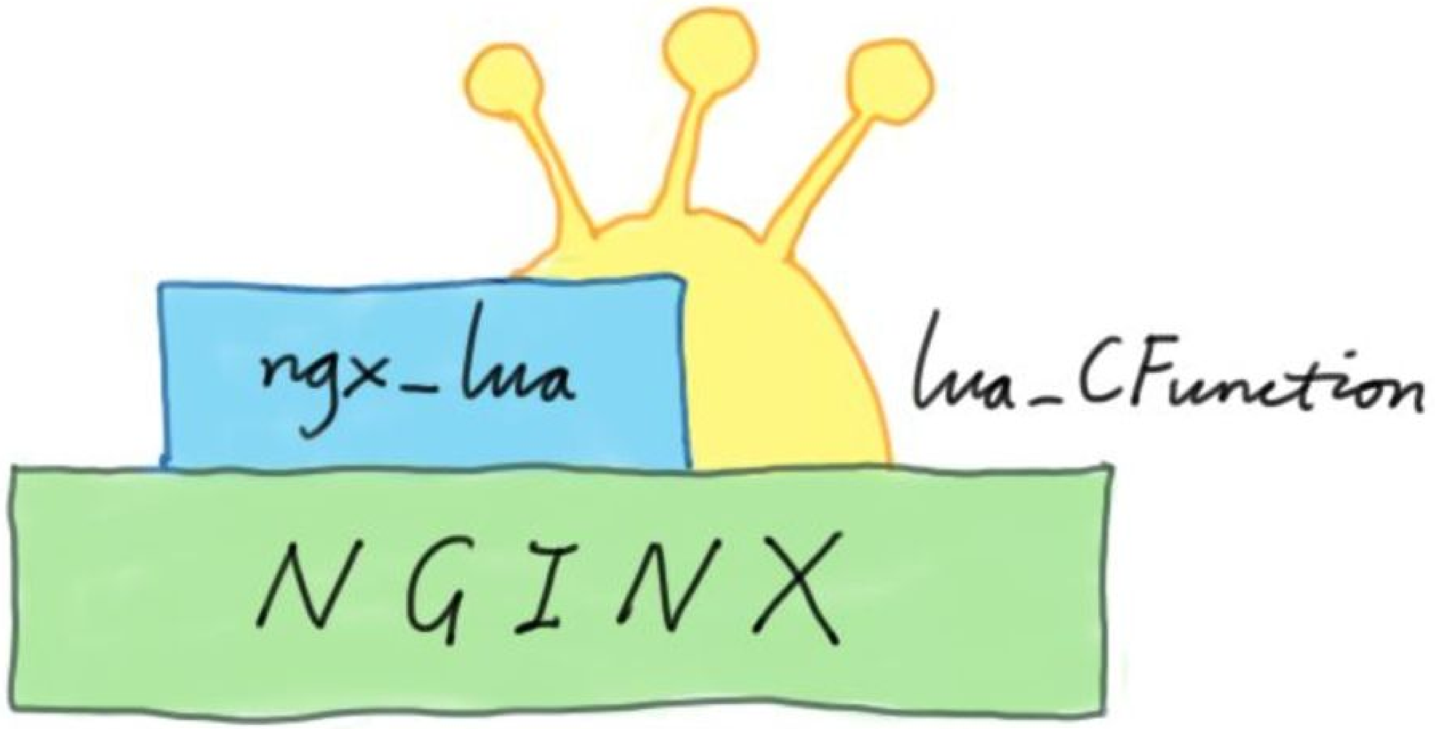
Lua functions

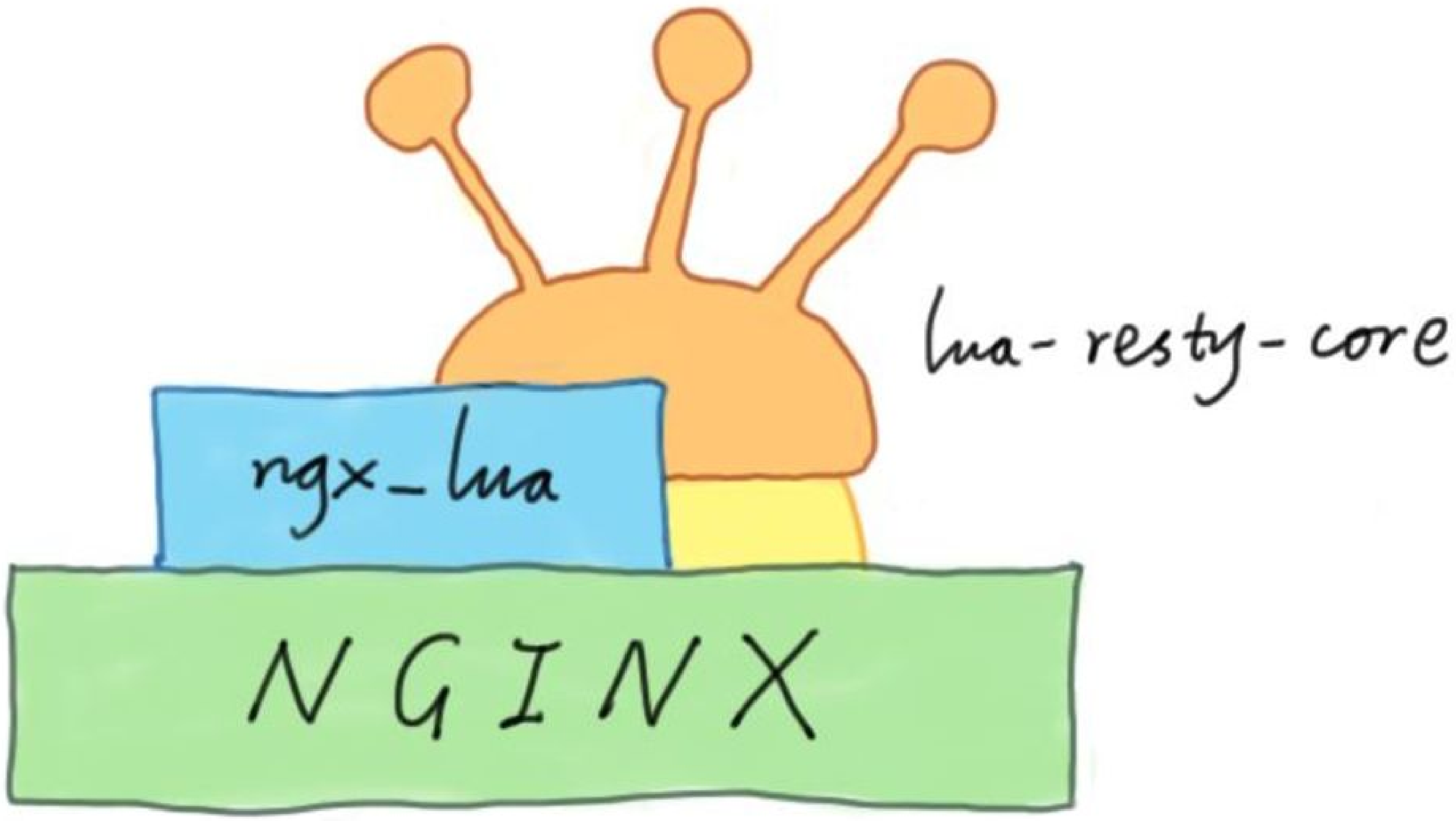
```
foo = function (...)
      ...
end
```

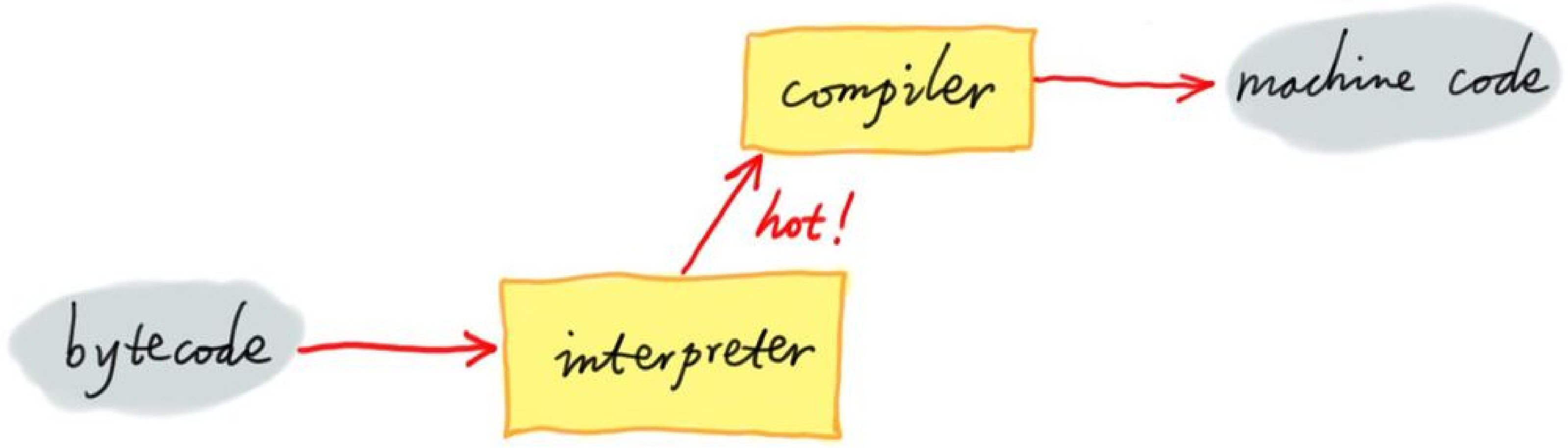

gc_swee..
gc_oneste..
lj_gc_ste..
lj_meta_cat
lj_func_newL_g..
lj_BC_CAT
lj_BC_FNEW
lj_BC
lua_resume
ngx_http_lua_run_thread

♡ JITting vs Interpreting

lua-resty-core


ngx_lua
lua_CFunction
NGINX


lua-resty-core
ngx_lua
NGINX


bytecode
interpreter
hot!
compiler
machine code

```
jit.v

jit.dump
```

```
lj-lua-stacks.sxx --arg nojit=1 ...

lj-lua-stacks.sxx --arg nointerp=1 ...
```

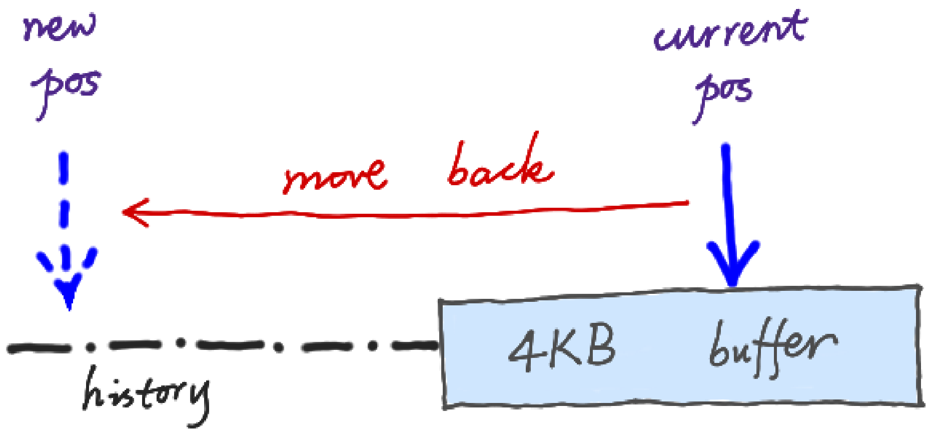
♡ *Biased* vs Unbiased Branching

♡ Lua code *generation* atop LuaJIT

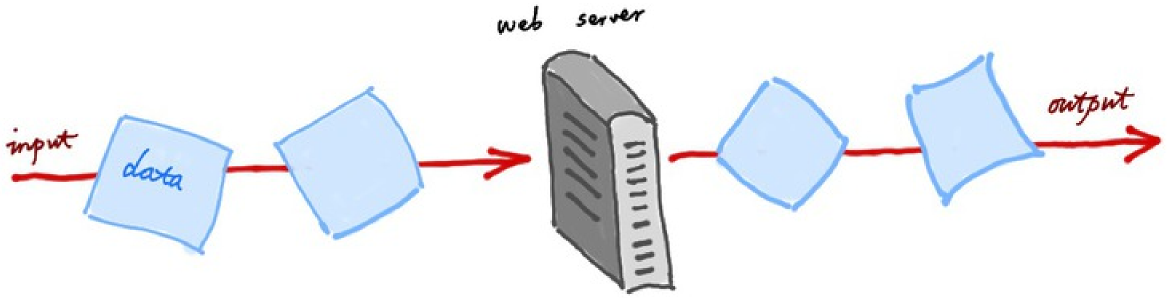
JIT over a JIT!

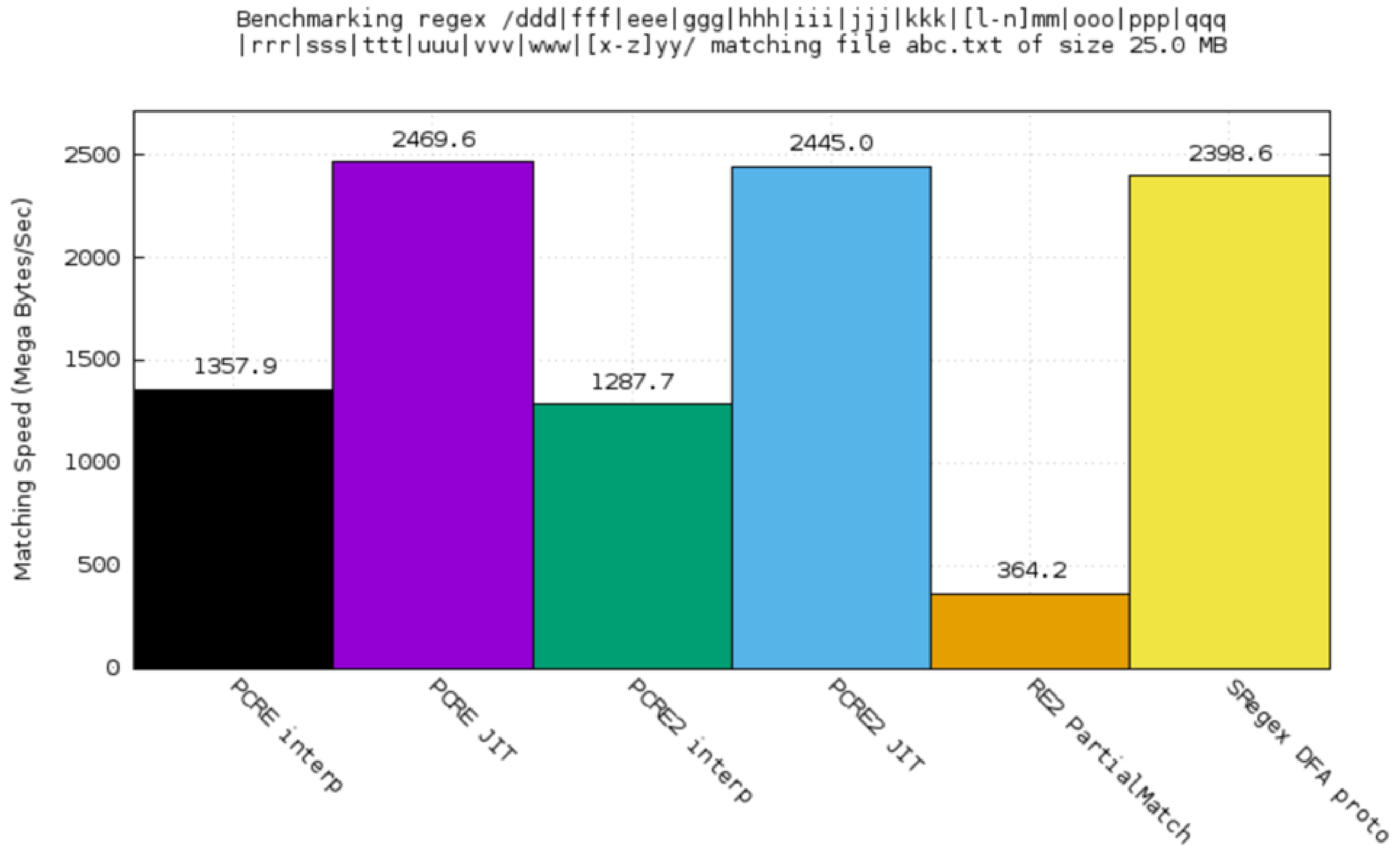
♡ Regexes

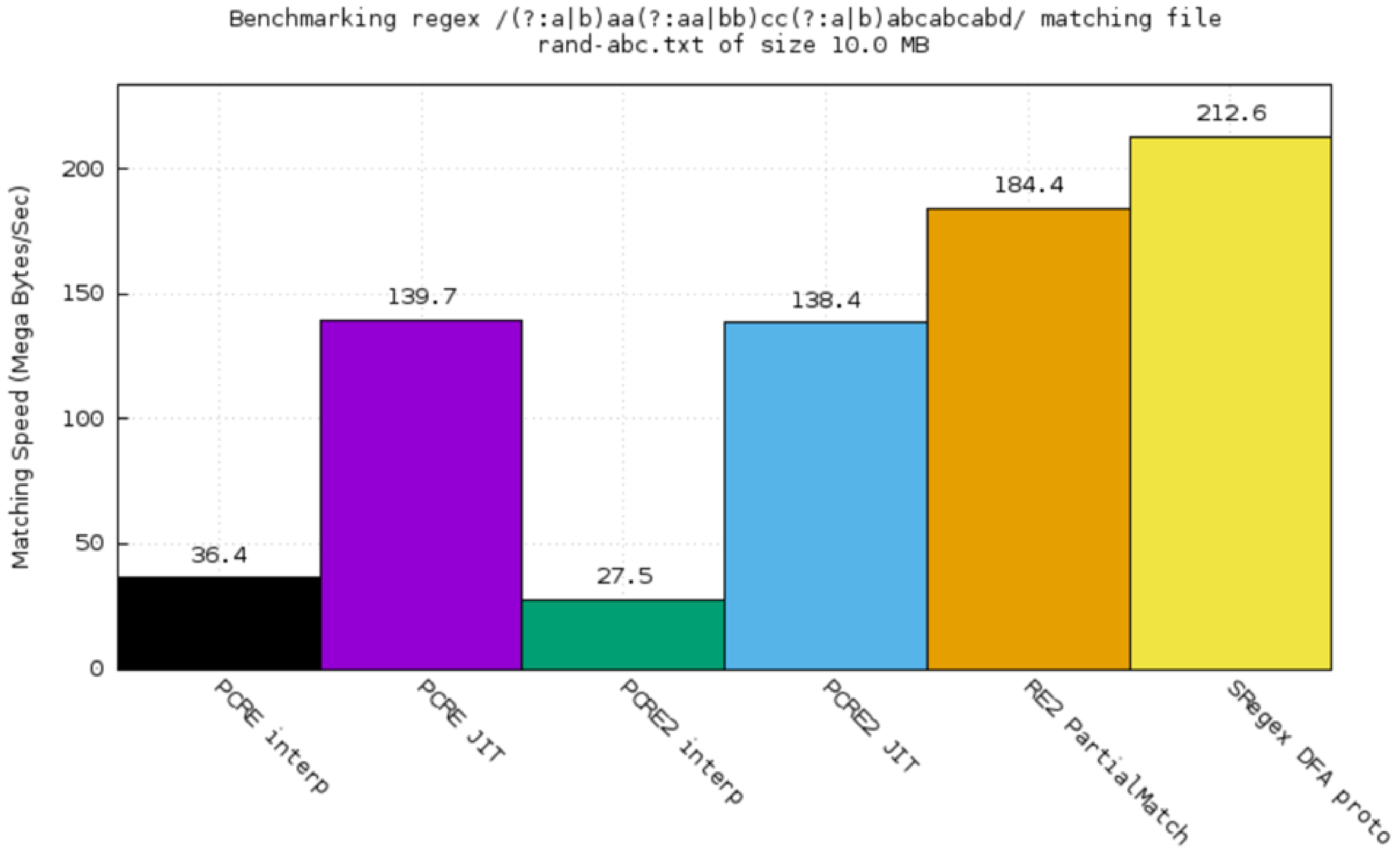
```
/
   \d+ \. \d+
| \. \d+
| \d+
/x
```


new pos
current pos
move back
4KB buffer
history

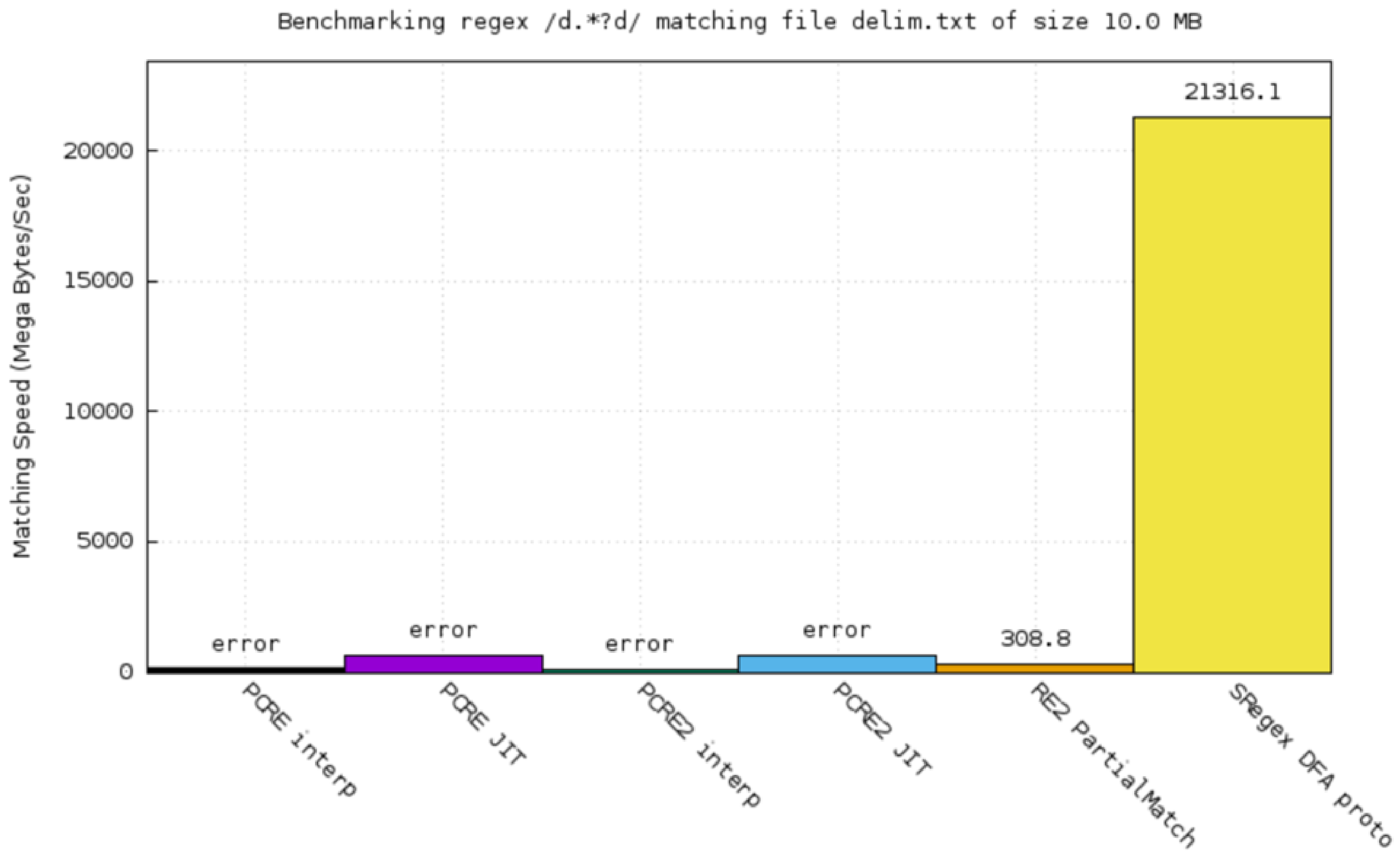
sregex


input
data
web server
output

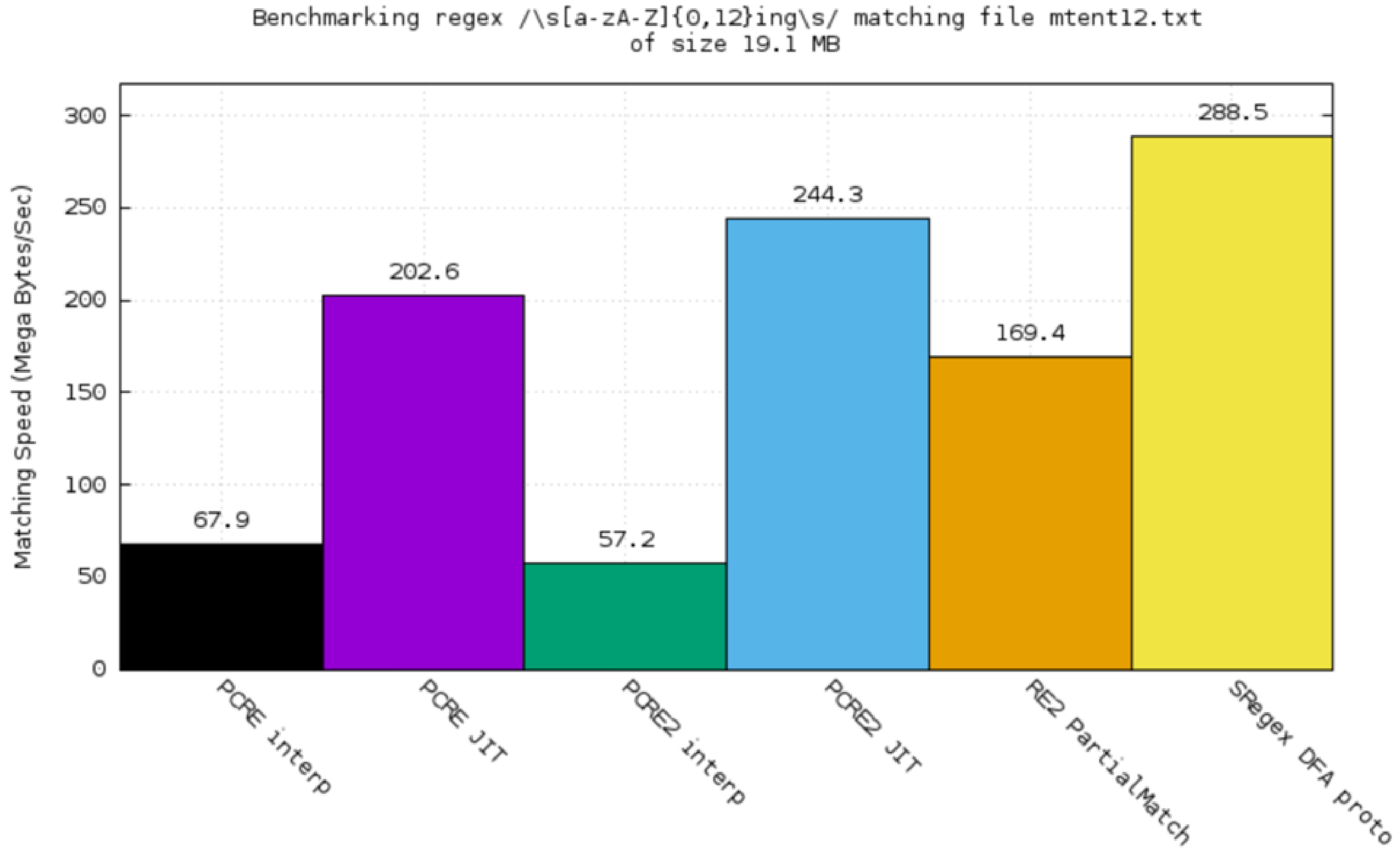

Benchmarking regex /ddd|fff|eee|ggg|hhh|iii|jjj|kkk|[l-n]mm|ooo|ppp|qqq
|rrr|sss|ttt|uuu|vvv|www|[x-z]yy/ matching file abc.txt of size 25.0 MB
Matching Speed (Mega Bytes/Sec)
0
500
1000
1500
2000
2500
1357.9
2469.6
1287.7
2445.0
364.2
2398.6
PCRE interp
PCRE JIT
PCRE2 interp
PCRE2 JIT
RE2 PartialMatch
SRegex DFA proto



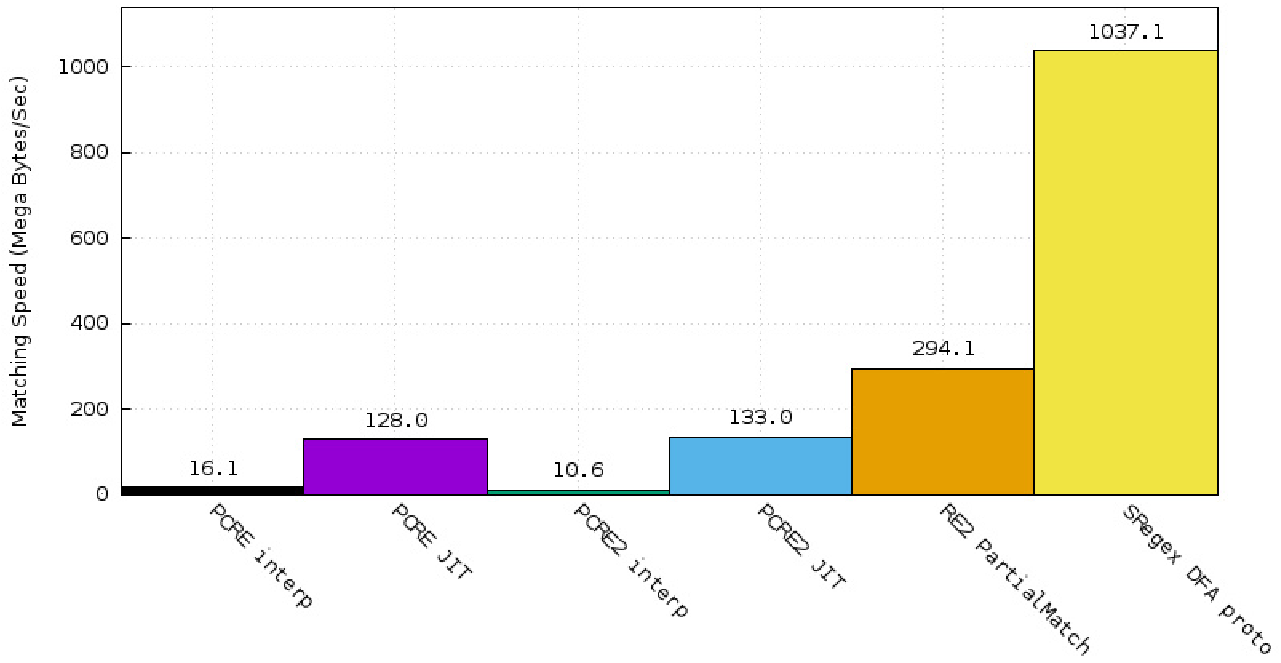
Benchmarking regex /(?:a|b)aa(?:aa|bb)cc(?:a|b)abcabcabd/ matching file rand-abc.txt of size 10.0 MB
Matching Speed (Mega Bytes/Sec)
0
50
100
150
200
36.4
139.7
27.5
138.4
184.4
212.6
PCRE interp
PCRE JIT
PCRE2 interp
PCRE2 JIT
RE2 PartialMatch
SRegex DFA proto



Benchmarking regex /d.*?d/ matching file delim.txt of size 10.0 MB
Matching Speed (Mega Bytes/Sec)
0
5000
10000
15000
20000
error
error
error
error
308.8
21316.1
PCRE interp
PCRE JIT
PCRE2 interp
PCRE2 JIT
RE2 PartialMatch
SRegex DFA proto

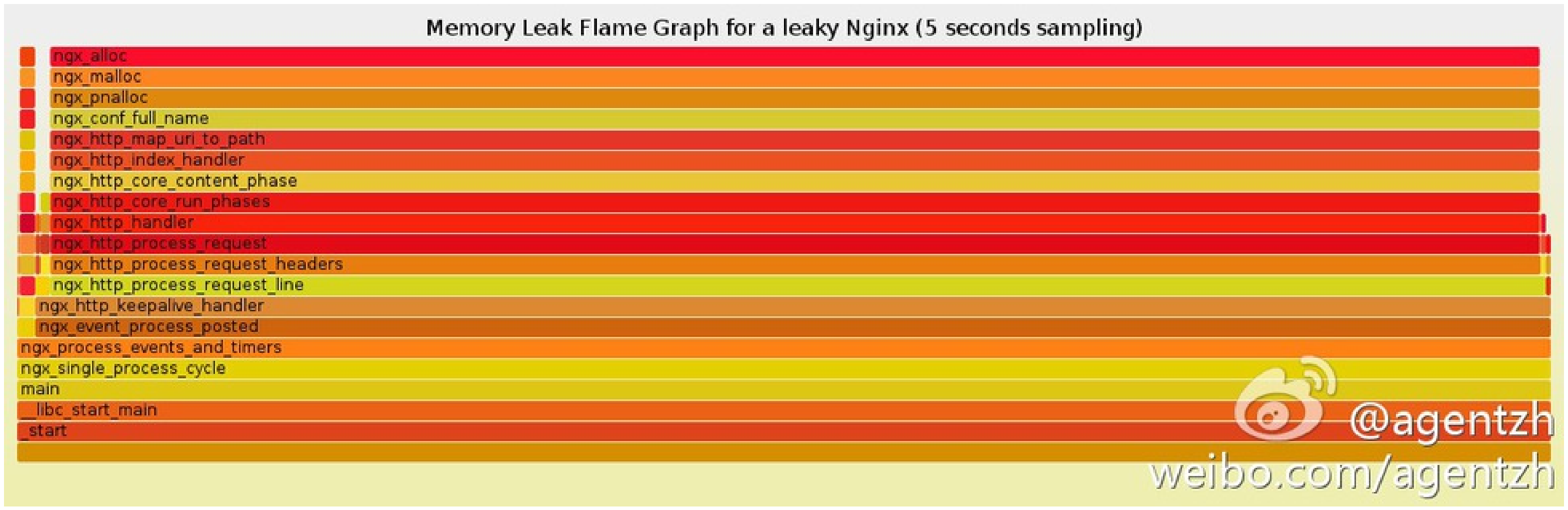

Benchmarking regex /\s[a-zA-Z]{0,12}ing\s/ matching file mtent12.txt
of size 19.1 MB
Matching Speed (Mega Bytes/Sec)
0
50
100
150
200
250
300
67.9
202.6
57.2
244.3
169.4
288.5
PCRE interp
PCRE JIT
PCRE2 interp
PCRE2 JIT
RE2 PartialMatch
SRegex DFA proto

Benchmarking regex /(?i)(?:merge.*?using\s*?\()|(execute\s*?immediate\s*?[\"'\x60´’‘
])|(?:\W\d*?\s*?having\s*?[^\s\-])|(?:match\s*?[\w(),+-]+\s*?against
\s*?\()/
matching file regex/misc/5/text.txt of size 0.0 MB



☺ Memory

♡ Memory-Leak Flame Graphs

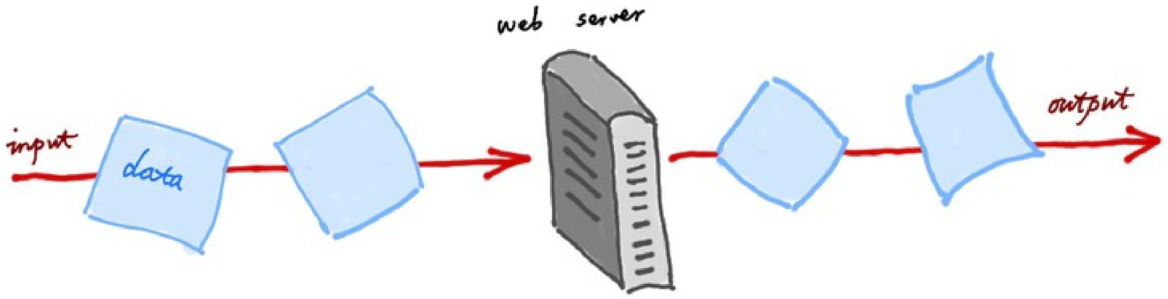

Memory Leak Flame Graph for a leaky Nginx (5 seconds sampling)
ngx_alloc
ngx_malloc
ngx_pnalloc
ngx_conf_full_name
ngx_http_map_uri_to_path
ngx_http_index_handler
ngx_http_core_content_phase
ngx_http_core_run_phases
ngx_http_handler
ngx_http_process_request
ngx_http_process_request_headers
ngx_http_process_request_line
ngx_http_keepalive_handler
ngx_event_process_posted
ngx_process_events_and_timers
ngx_single_process_cycle
main
__libc_start_main
_start
@agentzh
weibo.com/agentzh

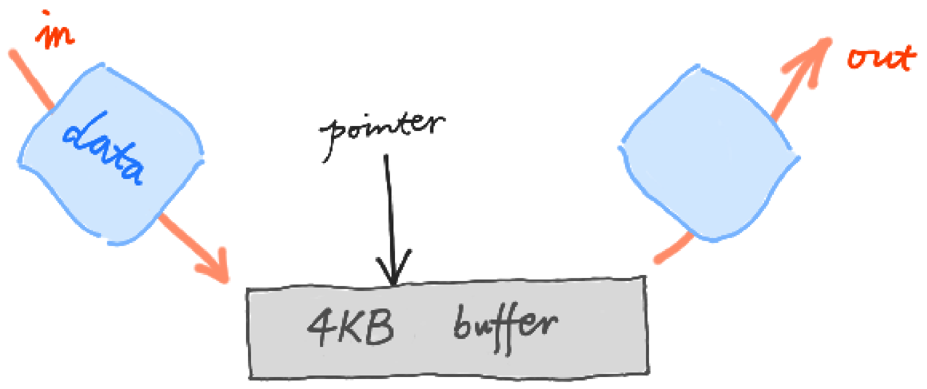
♡ GC Object Analaysis

```
$ lj-gc-objs.sxx -x 14378 -D MAXACTION=200000
Start tracing 14378 (/opt/nginx/sbin/nginx)

main machine code area size: 65536 bytes
C callback machine code size: 4096 bytes
GC total size: 9683407 bytes
GC state: pause

27948 table objects: max=131112, avg=106, min=32, sum=2983944 (in bytes)
22343 string objects: max=1421562, avg=198, min=18, sum=4432482 (in bytes)
12168 userdata objects: max=8916, avg=50, min=27, sum=619223 (in bytes)
2837 function objects: max=148, avg=27, min=20, sum=78264 (in bytes)
1200 upvalue objects: max=24, avg=24, min=24, sum=28800 (in bytes)
650 proto objects: max=3860, avg=313, min=74, sum=203902 (in bytes)
349 thread objects: max=1648, avg=774, min=424, sum=270464 (in bytes)
202 trace objects: max=1560, avg=375, min=160, sum=75832 (in bytes)
9 cdata objects: max=36, avg=17, min=12, sum=156 (in bytes)
JIT state size: 7696 bytes
global state tmpbuf size: 710772 bytes
C type state size: 4568 bytes

My GC walker detected for total 9683407 bytes.
45008 microseconds elapsed in the probe handler.
```

```
(gdb) lgcstat
15172 str        objects: max=2956, avg = 51, min=18, sum=779126
 987 upval      objects: max=24, avg = 24, min=24, sum=23688
 104 thread     objects: max=1648, avg = 1622, min=528, sum=168784
 431 proto      objects: max=226274, avg = 2234, min=78, sum=963196
 952 func       objects: max=144, avg = 30, min=20, sum=28900
 446 trace      objects: max=23400, avg = 1857, min=160, sum=828604
2965 cdata      objects: max=4112, avg = 17, min=12, sum=51576
18961 tab        objects: max=24608, avg = 207, min=32, sum=3943256
   9 udata      objects: max=176095, avg = 39313, min=32, sum=353822
```

♡ Streaming Processing


input
data
web server
output


in
data
pointer
out
4KB buffer

♡ Streaming Regex (sregex)

♡ The *cost* of abstractions

♡ The oppportunities of *new* abstractions

♡ Business-Level Domain Specific Languages

ModSecurity's syntax *sucks*.

☺ *Any questions*? ☺