Shoving Your Application Code Into the Web Server

Using OpenResty and nginx to write high-performance web applications

Nikola Plejić

WebCamp Zagreb, October 6th, 2018

Who Am I?

- dabbling around the web for a while
- GNU/Linux, Rust, Lua, C++, Clojure, Python
- senior engineer, TVbeat (of course we're hiring)

Who Am I?

- dabbling around the web for a while
- GNU/Linux, Rust, Lua, C++, Clojure, Python
- senior engineer, TVbeat (of course we're hiring)
- mostly HTTP

web server


NGINX

- “c10k” problem

- “c10k” problem
- 30%-40% of the public internet (*)

- “c10k” problem
- 30%-40% of the public internet (*)
  - (*) - lies, big lies, and statistics

- “c10k” problem
- 30%-40% of the public internet (*)
  - (*) - lies, big lies, and statistics
- modular architecture

- “c10k” problem
- 30%-40% of the public internet (*)
  - (*) - lies, big lies, and statistics
- modular architecture
- asynchronous, event-driven model

modularity

modularity

- LDAP -> PostgreSQL -> upstream

modularity

- LDAP -> PostgreSQL -> upstream
- append X-Requested-By header to every HTTP request

modularity

- LDAP -> PostgreSQL -> upstream
- append X-Requested-By header to every HTTP request
- REST -> Protobuf

what if we could script these nginx modules?

Lua

- small, dynamic programming language

Lua

- small, dynamic programming language
- think: JavaScript, Scheme, Tcl

Lua

- small, dynamic programming language
- think: JavaScript, Scheme, Tcl
- ANSI C: highly portable and embedable

Lua

- small, dynamic programming language
- think: JavaScript, Scheme, Tcl
- ANSI C: highly portable and embedable
- LuaJIT: probably the best dynamic language JIT compiler in the world (*)

Lua

- small, dynamic programming language
- think: JavaScript, Scheme, Tcl
- ANSI C: highly portable and embedable
- LuaJIT: probably the best dynamic language JIT compiler in the world (*)

(*) - lies, big lies, and benchmarks

Lua - example

```lua
function fact(n)
  if n == 1 then
    return 1
  end

  return n * fact(n - 1)
end

print(fact(128))
```

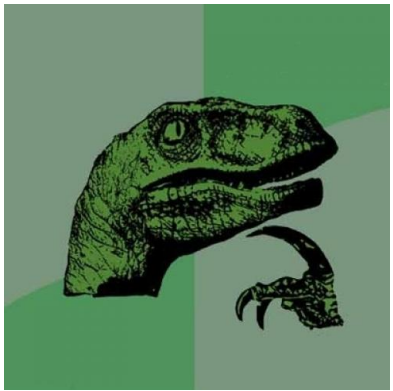
Lua - example

```lua
function allwords ()
  local line = io.read()
  local pos = 1
  return function ()
    while line do
      local s, e = string.find(line, "%w+", pos)
      if s then          -- found a word?
        pos = e + 1      -- update next position
        return string.sub(line, s, e)
      else
        line = io.read()-- word not found; try next line
        pos = 1          -- restart from first position
      end
    end
    return nil           -- no more lines: end of traversal
  end
end
```

"Programming in Lua", 1st ed. §10.2

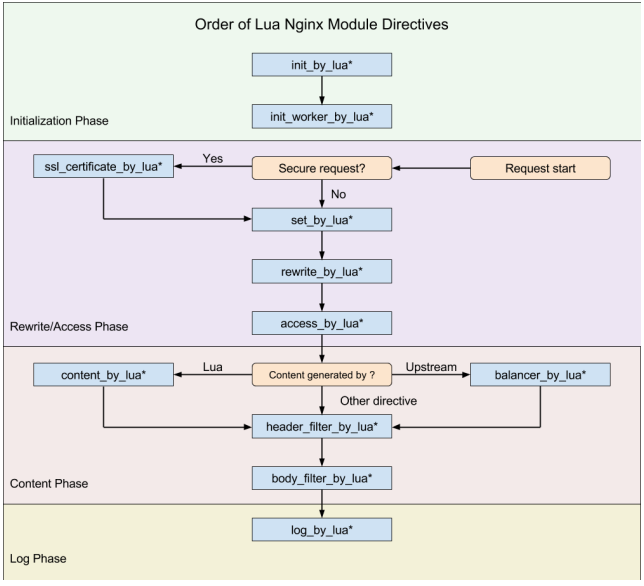
Lua - embedable in C apps...
nginx - written in C...

Lua - embedable in C apps...
nginx - written in C...



ngx_http_lua_module

ngx_http_lua_module



ngx_http_lua_module

```
worker_processes  1;
error_log logs/error.log;
events {
    worker_connections 1024;
}
http {
    server {
        listen 8080;
        location /brew {
            default_type text/html;
            content_by_lua_block {
              -- Lua code
            }
        }
    }
}
```

ngx_http_lua_module

content_by_lua_block:

```
ngx.req.read_body()
local body = ngx.req.get_body_data()

if body and (body:find("coffee") or
             body:find("beer")) then
  ngx.status = 418
  ngx.say("I'm a teapot")
elseif body and body:find("tea") then
  ngx.status = 200
  ngx.say("Coming right up")
else
  ngx.status = 400
  ngx.say("Confused")
end

return ngx.exit(ngx.status)
```

ngx_http_lua_module

```
http {
  upstream backend {
    server 0.0.0.1;

    balancer_by_lua_block {
      -- Lua code
    }
  }

  server {
    listen 8080;

    location / {
      proxy_pass http://backend/;
    }
  }
}
```

ngx_http_lua_module

balancer_by_lua_block:

```
local redis = require "resty.redis"
local balancer = require "ngx.balancer"

local r = redis:new()
local ok, err = r:connect("127.0.0.1", 6379)

local host, err = r:get("my_upstream:host")
local port, err = r:get("my_upstream:port")

local ok, err = balancer.set_current_peer(host, port)

if not ok then
  ngx.log(ngx.ERR, "failed to set peer: ", err)
  return ngx.exit(500)
end
```

OpenResty

OpenResty

nginx

OpenResty

nginx + LuaJIT

OpenResty

nginx + LuaJIT + async libraries

OpenResty

ngx_http_lua_module $\approx$ OpenResty

non-trivial usage

non-trivial usage

- loading code from files: content_by_lua_file

non-trivial usage

- loading code from files: content_by_lua_file
- LuaRocks is fair game

non-trivial usage

- loading code from files: content_by_lua_file
- LuaRocks is fair game
- testing: busted + Test::Nginx::Socket

non-trivial usage

- loading code from files: content_by_lua_file
- LuaRocks is fair game
- testing: busted + Test::Nginx::Socket
- profiler: a set of SystemTap-based scripts

what's not to like?

what's not to like?

- Lua fragmentation: LuaJIT vs. PUC-Lua-current

what's not to like?

- Lua fragmentation: LuaJIT vs. PUC-Lua-current
- it's not easy being asynchronous

what's not to like?

- Lua fragmentation: LuaJIT vs. PUC-Lua-current
- it's not easy being asynchronous
  - use OpenResty libraries
  - get your hands dirty with C

Turing-complete web server configuration

Thank you!

Thank you!

Rate the talk!
https://joind.in/talk/2de00

Thank you!

Rate the talk!
https://joind.in/talk/2de00

Questions?

nikola@plejic.com — @nikolap:matrix.org