NGINX SCRIPTING

EXTENDING NGINX FUNCTIONALITIES WITH LUA

Tony Fabeen / *@tonyfabeen* / *SlimStacks*

WHO AM I

slimstacks

NGINX ('ENGINE-X')

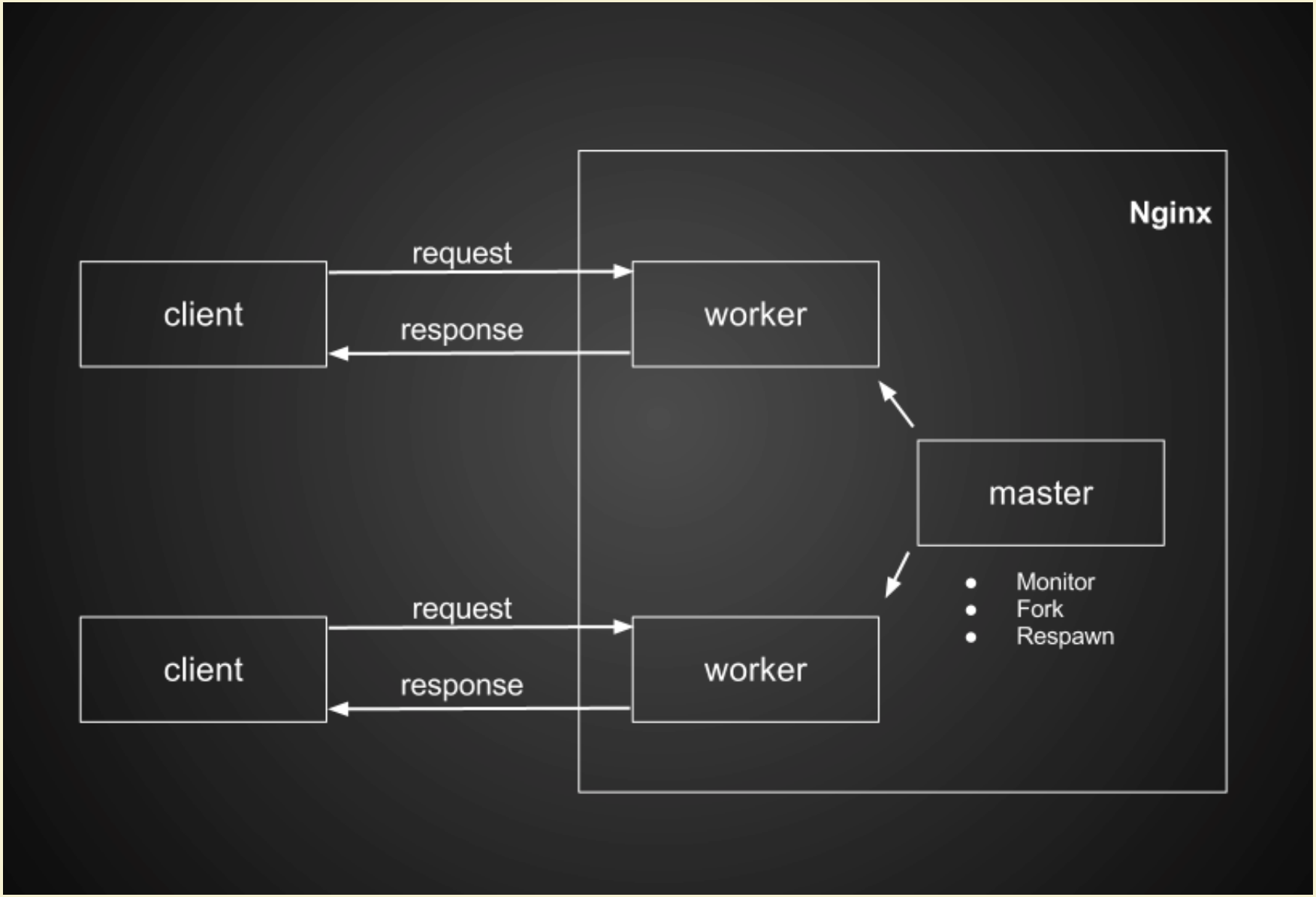
- High performance HTTP, POP/IMAP and reverse proxy server.
- Started in 2002 by Igor Sysoev, public in 2004.
- Entirely written in C.
- Hosts nearly 12.18% of active sites across all domains.
- Nginx.com in 2011.

MASTER WORKER MODEL

```
$ ps aux | grep nginx

root 31329 ... nginx: master process /opt/nginx/sbin/nginx

www  31330 ... nginx: worker process
```


Nginx
request
client
response
worker
master
Monitor
Fork
Respawn
request
client
response
worker

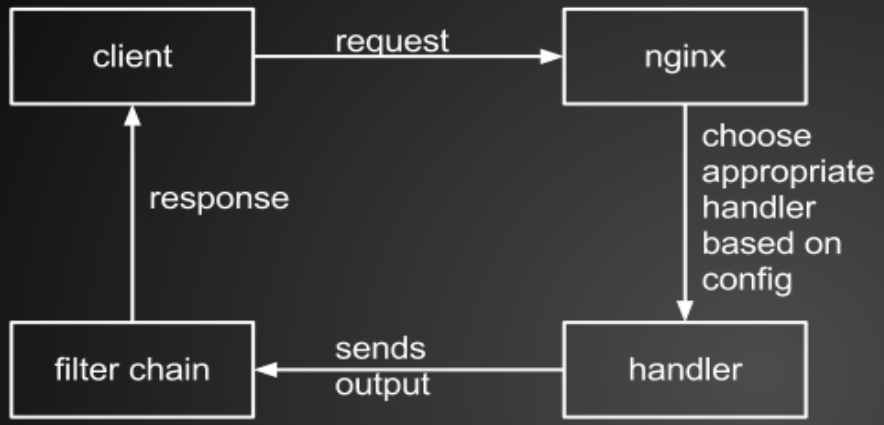
MASTER PROCESS

- reading and validating configuration
- creating, binding and closing sockets
- starting, terminating and maintaining the configured number of `worker` processes
- re-opening log files
- compiling embedded Perl scripts

WORKER PROCESS

- Do all important stuff
- Handle connection from clients
- Reverse Proxy and Filtering functionalities

REQUEST PROCESSING



Request Phases

- server rewrite
- location
- location rewrite
- access control
- try_files
- log

config file

```
...

location /admin {
  root /var/www/admin;
  mymodule on;
}

...

upstream fastcgi {
  server 127.0.0.1:9000;
  server 127.0.0.1:9001;
}
location ~* \.php$ {
  fastcgi_pass fastcgi;
}
```

REQUEST PHASES

SERVER REWRITE PHASE

request URI transformation on virtual server level

FIND CONFIG PHASE

configuration location lookup

REWRITE PHASE

request URI transformation on location level

ACCESS PHASE

access restrictions check phase

TRY FILES PHASE

try_files directive processing phase

CONTENT PHASE

content generation phase

LOG PHASE

logging phase

MODULARITY

- Core Module
- Functional Modules

CORE MODULE

- Event Loop
- Module execution control

FUNCTIONAL MODULES

- Read from / Write to Network and Storage
- Transform Content
- Outbound Filtering
- Server Side Includes
- Upstream Server communication
- ...etc

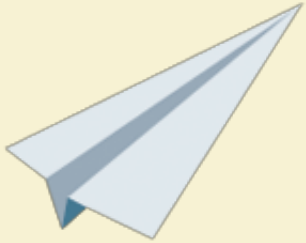
LUA ON THE STAGE

A BIT OF LUA

- Created in Brazil
- Portable
- Simple
- Small
- Easy to embed
- Fast

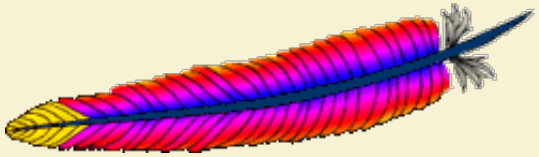
OSS USING LUA

redis



LIGHTTPD

fly light.



LUA NGINX MODULE

https://github.com/chaoslawful/lua-nginx-module/

- Created by **TaoBao.com** Engineers
- High concurrent and non-blocking request processing
- Programs can be written in the plain-old sequential way
- Nginx takes care of I/O operations and Lua Nginx Module restore the context and resume the program logic

LUA NGINX MODULE

https://github.com/chaoslawful/lua-nginx-module

- Introduces directives for running Lua inside Nginx
- Exposes the Nginx environment to Lua via an Api
- It's fast
- Is even faster when compiled with LUA JIT(Just in Time Compiler)

NGINX LUA API

DIRECTIVES

Configuration directives serve as gateways to the Lua API within the nginx.conf file.

- `content_by_lua` **`LUA_SCRIPT_STRING`**
- `rewrite_by_lua` **`LUA_SCRIPT_STRING`**
- `access_by_lua` **`LUA_SCRIPT_STRING`**
- `content_by_lua_file` **`PATH_TO_LUA_SCRIPT_FILE`**
- `rewrite_by_lua_file` **`PATH_TO_LUA_SCRIPT_FILE`**
- `access_by_lua_file` **`PATH_TO_LUA_SCRIPT_FILE`**

Unless you set **`lua_code_cache`** to **`off`** , modules will be loaded once on the first request.

NGX PACKAGE

Nginx Environment is exposed via **`ngx`** package

- `ngx.arg.`**`url_arg`**
- `ngx.var.`**`VARIABLE_NAME`**
- `ngx.header.`**`HEADER_ATTRIBUTE`**
- `ngx.ctx`

HELLO WORLD !

```
    location /hello-user-by-lua {
      default_type "text/plain";
      content_by_lua '
        ngx.say("Hello, ", ngx.var.arg_name, "!")
      ';
    }

    location /hello-user-by-nginx {
      echo "Hello, $arg_name !";
    }

$ curl http://localhost/hello-user-by-lua?name=DevInSampa
  Hello, DevInSampa !
$ curl http://localhost/hello-user-by-nginx?name=DevInSampa
  Hello, DevInSampa !
```

NGINX VARS

```
        location /acessing-nginx-args {
          set $first 35;
          set $second 65;

          set_by_lua $sum '
            return ngx.var.first + ngx.var.second
          ';

          echo "The sum is $sum";
        }

$ curl http://localhost/acessing-nginx-args
  The sum is 99
```

NGINX SUBREQUESTS

```
location /lua-subrequests {
  content_by_lua '
    local response = ngx.location.capture("/hello-user-by-nginx?name=
DevInSampa")
    if response.status >= 500 then
       ngx.exit(response.status)
    end

    ngx.status = response.status
    ngx.say(response.body)
  ';
}

$ curl http://localhost/lua-subrequests
  Hello, DevInSampa !
```

NON BLOCKING I/O SUBREQUESTS

```
location /analytics-increment {
    content_by_lua '
      local response = ngx.location.capture("/redis",
          {args = {cmd = "incr", key = ngx.var.arg_link}})
      ngx.say("Incremented to :", ngx.var.arg_link)
    ';
}
location /redis {
  internal;
  set_unescape_uri $key $arg_key;
  set_unescape_uri $cmd $arg_cmd;
  redis2_query $cmd $key;
  redis2_pass 127.0.0.1:6379;
}

$ curl http://localhost/analytics-increment?link=http://www.devinsampa.com.br
  Incremented to :http://www.devinsampa.com.br
```

FILTERS

HEADER FILTERS

```
location / {
  proxy_pass http://localhost:8080;
  header_filter_by_lua 'ngx.header.Server = "My Little Server"';
}

$ curl -i -X HEAD http://localhost/header-filter
  HTTP/1.1 200 OK
  Date: Sun, 09 Sep 2012 21:18:11 GMT
  Server: My Little Server
  Content-Type: text/html;charset=utf-8
  Content-Length: 449
  Connection: keep-alive
  Status: 200 OK
  X-Frame-Options: sameorigin
  X-XSS-Protection: 1; mode=block
  X-Cascade: pass
  X-Rack-Cache: miss
```

BODY FILTERS

```
location /body-filter {
  echo "My content";

  body_filter_by_lua '
    ngx.arg[1] = string.gsub(ngx.arg[1], "My", "Your")
    ngx.arg[2] = true --set eof or last chain buffer
  ';
}

$ curl http://localhost/body-filter
  Your content
```

COSOCKETS

- Non Blocking, of course
- Communicate via TCP or Unix domain sockets
- Keepalive mechanism avoid connect/close for each request

COSOCKETS

```
location /memcached-from-lua {
  content_by_lua '
    local sock = ngx.socket.connect("127.0.0.1", 11211)
    local bytes, err = sock:send("set foo bar\r\n")

    if not bytes then
      ngx.say("failed to send.. \n", err)
      return
    end

    local data = sock:receive()
    if not data then
      ngx.say("Failed to receive data..\n")
    end

    ngx.say("Result : ", data)
  ';

}
```

SOME LIBRARIES USING PURE COSOCKETS

- https://github.com/agentzh/lua-resty-memcached
- https://github.com/agentzh/lua-resty-redis
- https://github.com/agentzh/lua-resty-mysql

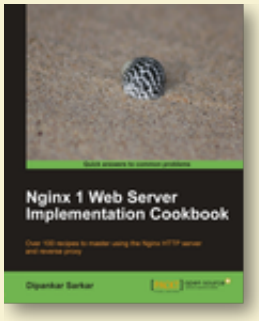
SUMMARY

- The Nginx architecture is excellent for highly scalable applications.
- Nginx can do a variety of things thanks to module extensions, and one can resuse those extensions by issuing sub-requests in Lua.
- lua-nginx-module makes use of the evented architecture in Nginx, providing a powerful and performant programming environment.
- It's possible to do 100% non-blocking I/O with readable code.

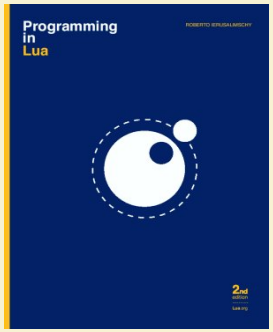
REFERENCES

- http://www.aosabook.org/en/nginx.html
- http://openresty.org
- http://www.evanmiller.org/nginx-modules-guide.html
- http://wiki.nginx.org/HttpLuaModule

BOOKS


Nginx 1 Web Server
Implementation Cookbook
Dipankar Sarkar


Nginx HTTP Server
PACKT


Programming
in
Lua
2nd

QUESTIONS

?

THANKS

http://www.twitter.com/tonyfabeen

http://www.linkedin.com/in/tonyfabeen

https://github.com/tonyfabeen