mod_wombat

Apache2 + Lua + Worker For The Win!

Brian McCallister
Ning

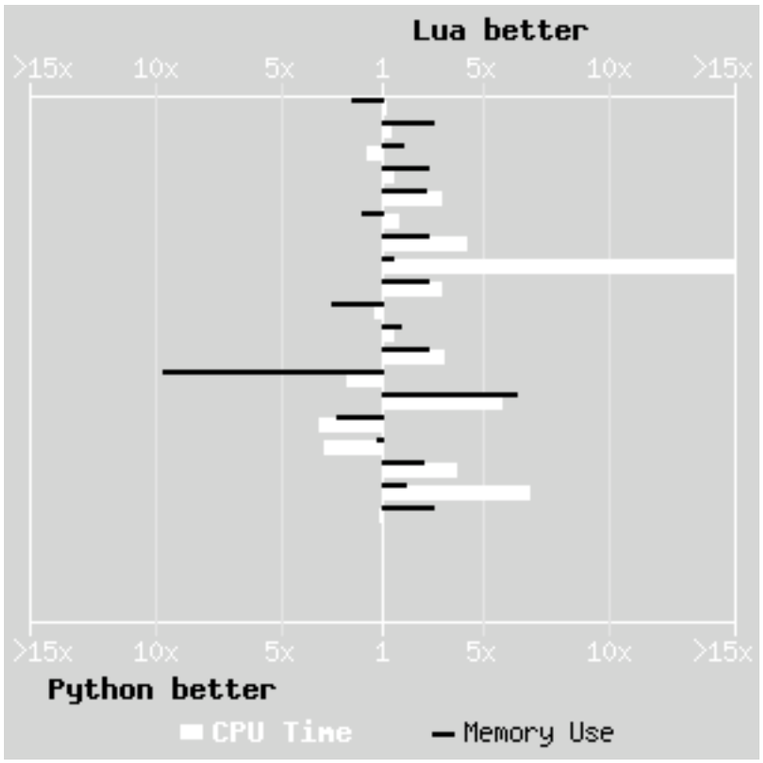
```
require 'apache2'
require 'math'

s = r:document_root() .. "%d.html"

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = s:format(math.random(5))
    r:debug("sending " .. r.filename)
    return apache2.OK
  end
  return apache2.DECLINED
end
```

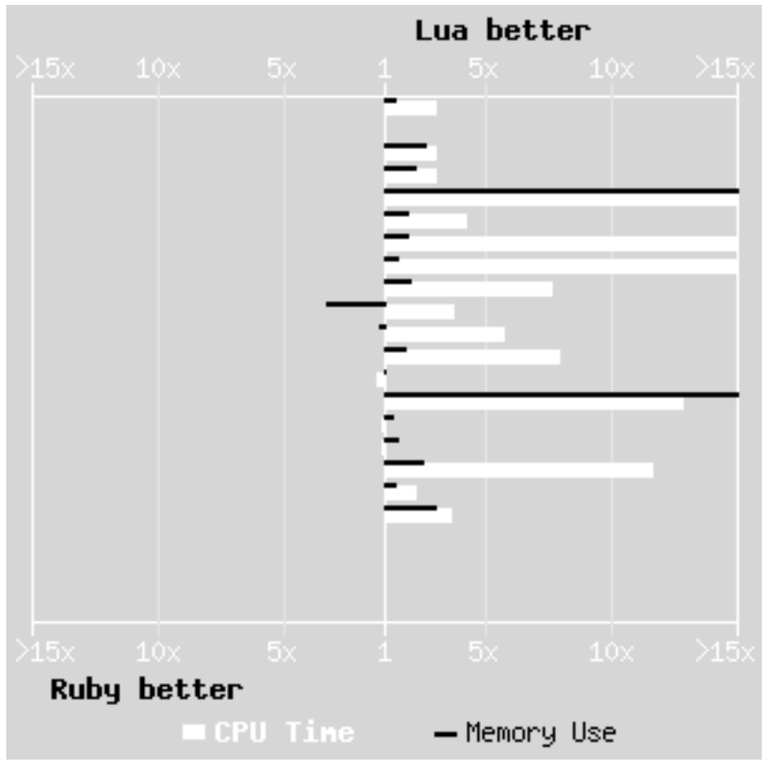
Why Lua?

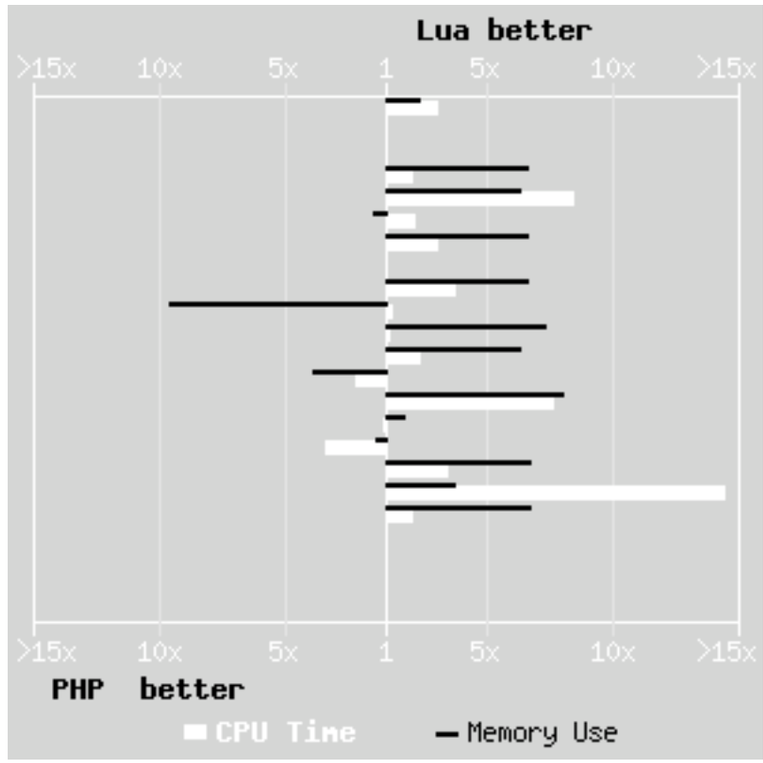
Threads

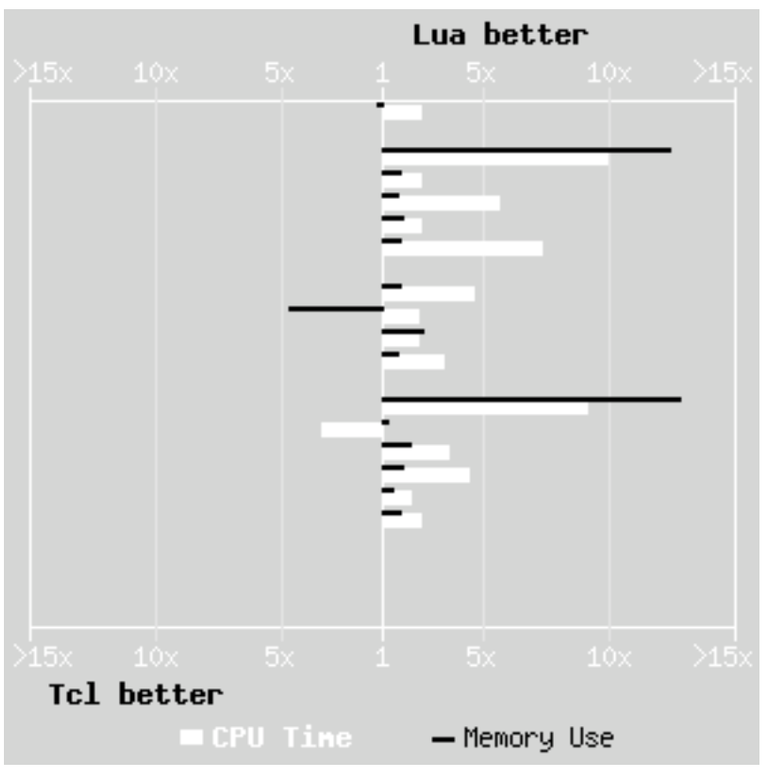
(cc) verybigjen http://flickr.com/photos/verybigjen/127593826/

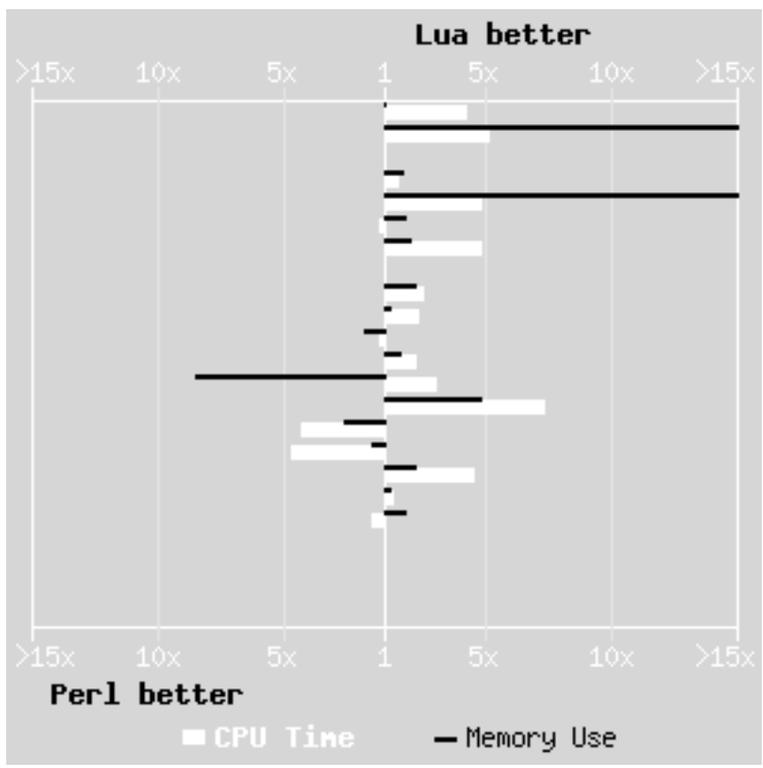
Performance

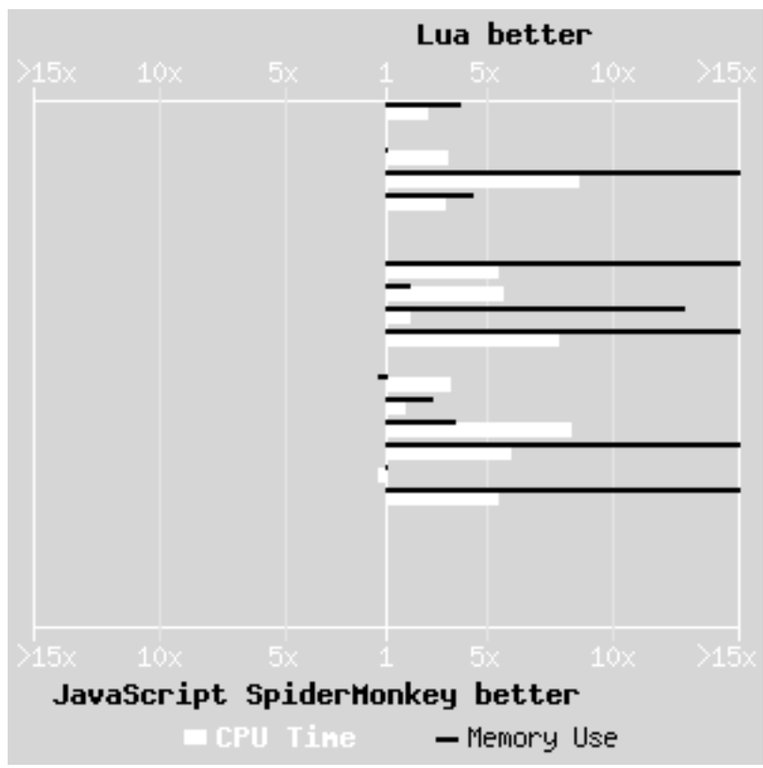












http://shootout.alioth.debian.org/gp4/benchmark.php

Overhead

Server	Reqs / Second	Notes
Apache 2.2 mod_hello_world	18,823.58	minimal module which just prints hello world
Apache 2.2 mod_wombat	17,856.76	server-scoped mod_wombat handler
Apache Tomcat 5.5	17,644.40	JSP
Jetty 6.1	12,449.36	JSP
Mongrel	2,378.05	HttpHandler, not Rails

http://kasparov.skife.org/blog/src/ruby/silly-micro-benchmarks.html

Working With C

```
static int lua_table_get(lua_State* L) {
    apr_table_t *t = check_apr_table(L, 1);
    const char* key = luaL_checkstring(L, 2);
    const char *val = apr_table_get(t, key);
    lua_pushstring(L, val);
    return 1;
}
```

Why Not $LANG ?

Guile

Gauche

Chicken

MZScheme

Elk

Scheme48

Python

SpiderMonkey

Perl

Ruby

TCL

PHP

Global Interpreter

Global Namespace

Thread Behavior

Size of Runtime

Line Noise :-)

Putting Wombat to Work

```
function bar(self, val)
  print(self.one .. " " .. val)
end

local foo = {
  one = 1,
  baz = bar
}

print(foo.one)
print(foo['one'])

foo:baz("hello")
foo.baz(foo, "hello")
bar(foo, "hello")
```

```
require 'apache2'
require 'math'

s = r:document_root() .. "%d.html"

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = s:format(math.random(5))
    r:debug("sending " .. r.filename)
    return apache2.OK
  end
  return apache2.DECLINED
end
```

```
require 'apache2'
require 'math'

s = r:document_root() .. "%d.html"

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = s:format(math.random(5))
    r:debug("sending " .. r.filename)
    return apache2.OK
  end
  return apache2.DECLINED
end
```

```
require 'apache2'
require 'math'

s = r:document_root() .. "%d.html"

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = s:format(math.random(5))
    r:debug("sending " .. r.filename)
    return apache2.OK
  end
  return apache2.DECLINED
end
```

```
require 'apache2'
require 'math'

s = r:document_root() .. "%d.html"

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = s:format(math.random(5))
    r:debug("sending " .. r.filename)
    return apache2.OK
  end
  return apache2.DECLINED
end
```

```
struct request_rec {
    /* ... */
    char *uri;
    char *filename;
    /* ... */
}
```

```
ServerRoot "/opt/httpd-2.2.10"
DocumentRoot "/www/ac_us_08/htdocs"
Listen 80

LoadModule apreq_module modules/mod_apreq2.so
LoadModule wombat_module modules/mod_wombat.so

LuaRoot "/www/ac_us_08/"

DirectoryIndex index.lua

AddHandler lua-script .lua

LuaHookTranslateName lib/hooks.lua \
                     rewrite_index
```

```
ServerRoot "/opt/httpd-2.2.10"
DocumentRoot "/www/ac_us_08/htdocs"
Listen 80

LoadModule apreq_module modules/mod_apreq2.so
LoadModule wombat_module modules/mod_wombat.so

LuaRoot "/www/ac_us_08/"

DirectoryIndex index.lua

AddHandler lua-script .lua

LuaHookTranslateName lib/hooks.lua \
                     rewrite_index
```

```
ServerRoot "/opt/httpd-2.2.10"
DocumentRoot "/www/ac_us_08/htdocs"
Listen 80

LoadModule apreq_module modules/mod_apreq2.so
LoadModule wombat_module modules/mod_wombat.so

LuaRoot "/www/ac_us_08/"

DirectoryIndex index.lua

AddHandler lua-script .lua

LuaHookTranslateName lib/hooks.lua \
                     rewrite_index
```

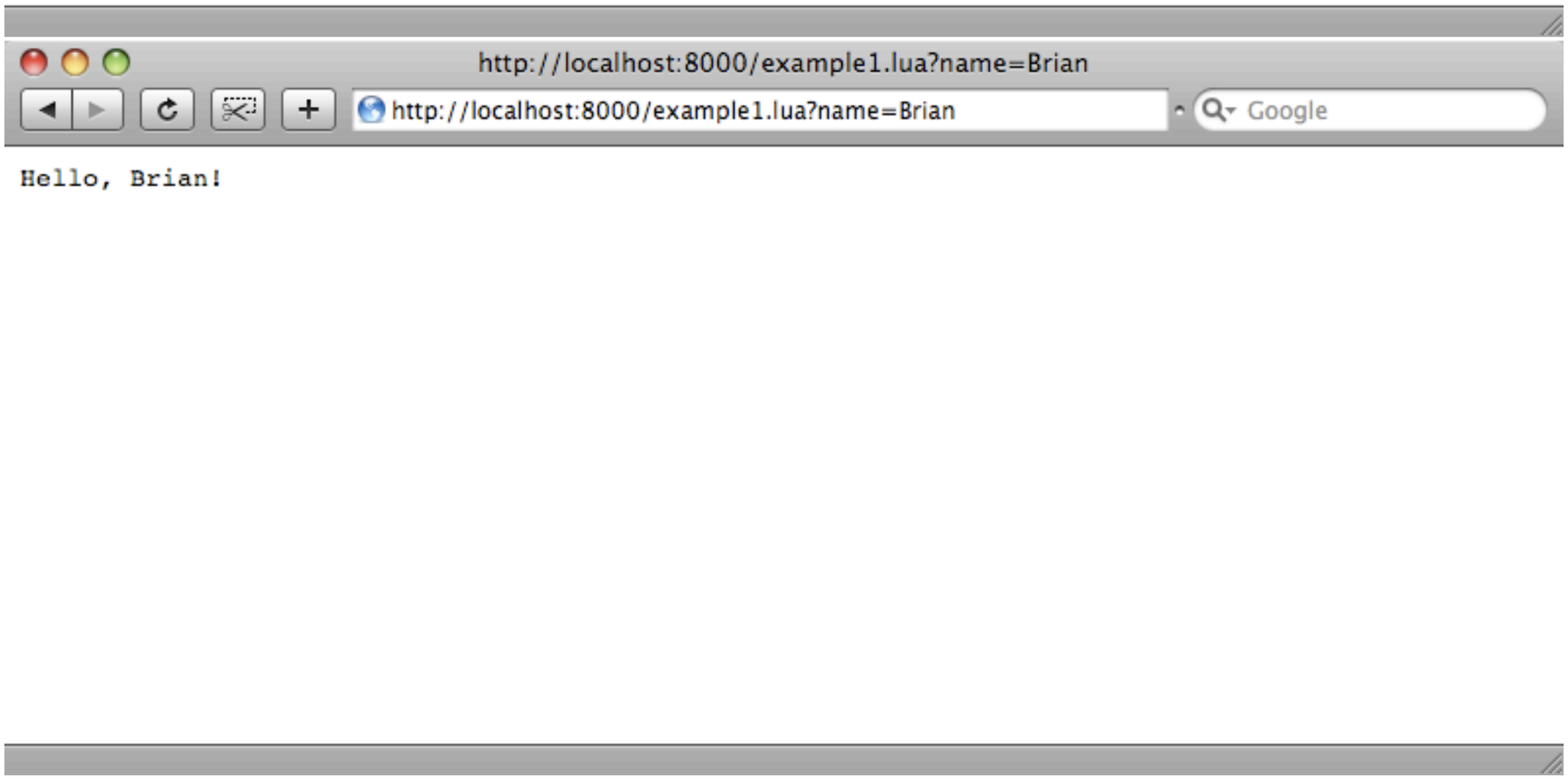
Evolving!

```
-- /example1.lua?name=Brian
require 'string'

function handle(r)
  local args = r:parseargs()
  if args['name'] then
    name = args.name
  else
    name = 'World'
  end
  r:puts( ('Hello, %s!\n'):format(name) )
end
```

Evolving!

```
-- /example1.lua?name=Brian
require 'string'

function handle(r)
   local args = r:parseargs()
   if args['name'] then
      name = args.name
   else
      name = 'World'
   end
   r:puts( ('Hello, %s!\n'):format(name) )
end
```


http://localhost:8000/example1.lua
http://localhost:8000/example1.lua
Google
Hello, World!

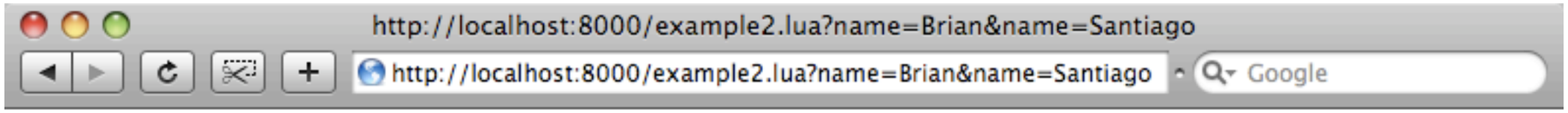

http://localhost:8000/example1.lua?name=Brian
http://localhost:8000/example1.lua?name=Brian
Google
Hello, Brian!

Evolving!

```
-- /example2.lua?name=Brian&name=Santiago

function handle(r)
  local _, full = r:parseargs()
  local names = full['name']

  for _, name in ipairs(names) do
    r:puts( ('Hello, %s!\n'):format(name) )
  end
end
```

Evolving!

```
-- /example2.lua?name=Brian&name=Santiago

function handle(r)
  local _, full = r:parseargs()
  local names = full['name']

  for _, name in ipairs(names) do
    r:puts( ('Hello, %s!\n'):format(name) )
  end
end
```




```
Hello, Brian!
Hello, Santiago!
```

```
function logging_stuff(r)
    r:debug("This is a debug log message")
    r:info("This is an info log message")
    r:notice("This is an notice log message")
    r:warn("This is an warn log message")
    r:err("This is an err log message")
    r:alert("This is an alert log message")
    r:crit("This is an crit log message")
    r:emerg("This is an emerg log message")
end
```

```
function server_rec(r)
  local server = r.server -- server_rec
  r:puts(server.server_hostname)
end
```

```
struct server_rec {
  /* ... */
  char *server_hostname;
  /* ... */
}
```

Evolving!

```
LuaRoot "/www/ac_us_08/"

LuaQuickHandler lib/hooks.lua quick
LuaHookTranslateName lib/hooks.lua trans_name
LuaHookMapToStorage lib/hooks.lua mapstorage
LuaHookAccessChecker lib/hooks.lua hosty
LuaHookCheckUserID lib/hooks.lua authn
LuaHookAuthChecker lib/hooks.lua authz
LuaHookTypeChecker lib/hooks.lua typecheck
LuaHookFixups lib/hooks.lua fixup

AddHandler lua-script .lua

MapLuaHandler ^/(\w+)_(\w+)$ lib/$1.lua h_$2
```

On the C Side

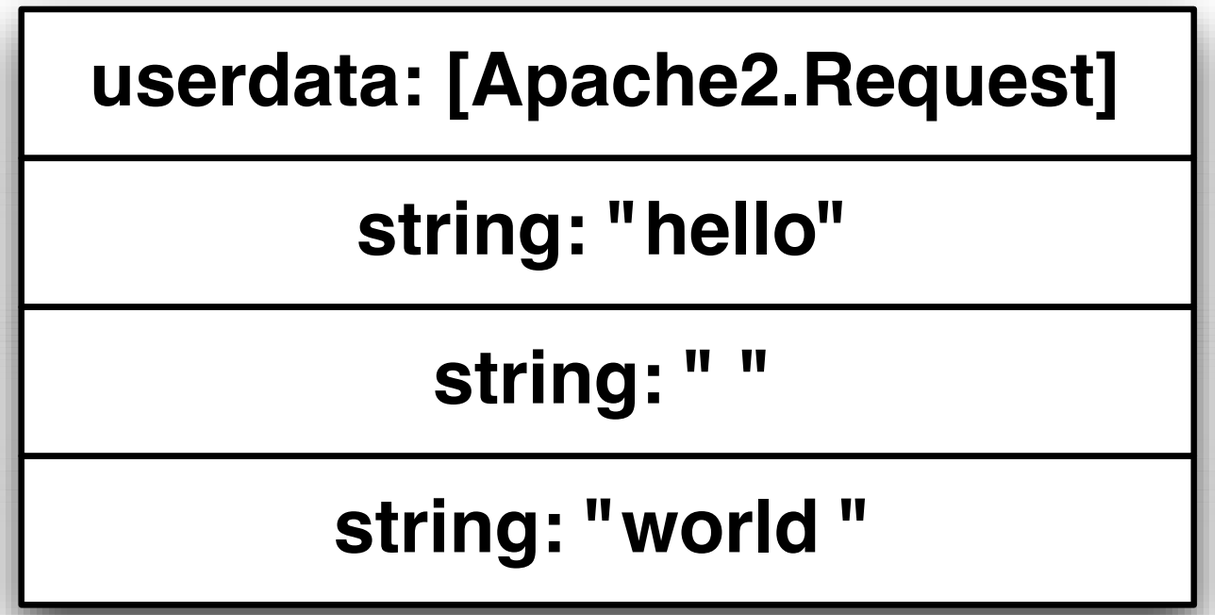
(Briefly)

```
require "myputs"

function handle(r)
  local msg = r:myputs("Hello", " ", "world")
  r:debug(msg)
end
```

```
require "myputs"

function handle(r)
  local msg = r:myputs("Hello", " ", "world")
  r:debug(msg)
end
```

```
r:myputs("Hello", " ", "world")
```

```
r.myputs(r, "Hello", " ", "world")
```

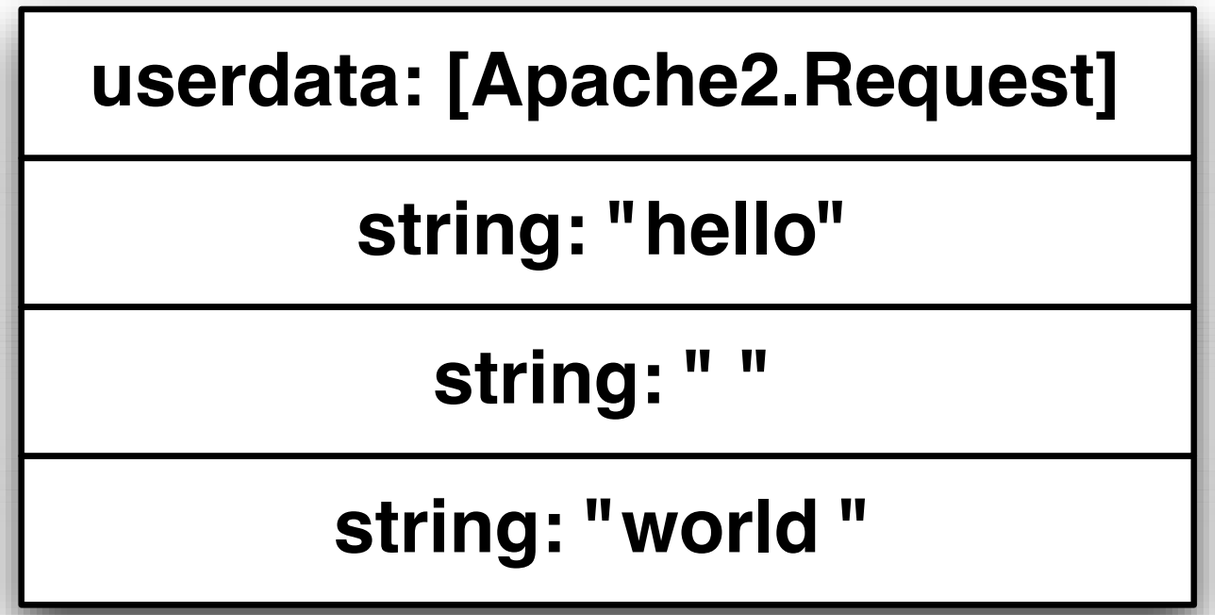

userdata: [Apache2.Request]
string: "hello"
string: " "
string: "world "

```
static int my_special_puts(lua_State* L) {
  request_rec* r = check_request_rec(L, 1);

  int argc = lua_gettop(L);
  int i;
  for (i=2;i<=argc;i++) {
    ap_rputs(luaL_checkstring(L, i), r);
  }
  lua_pushstring(L, "Thank You!");
  return 1;
}
```

```
static int my_special_puts(lua_State* L) {
  request_rec* r = check_request_rec(L, 1);

  int argc = lua_gettop(L);
  int i;
  for (i=2;i<=argc;i++) {
    ap_rputs(luaL_checkstring(L, i), r);
  }
  lua_pushstring(L, "Thank You!");
  return 1;
}
```

(lua_CFunction)

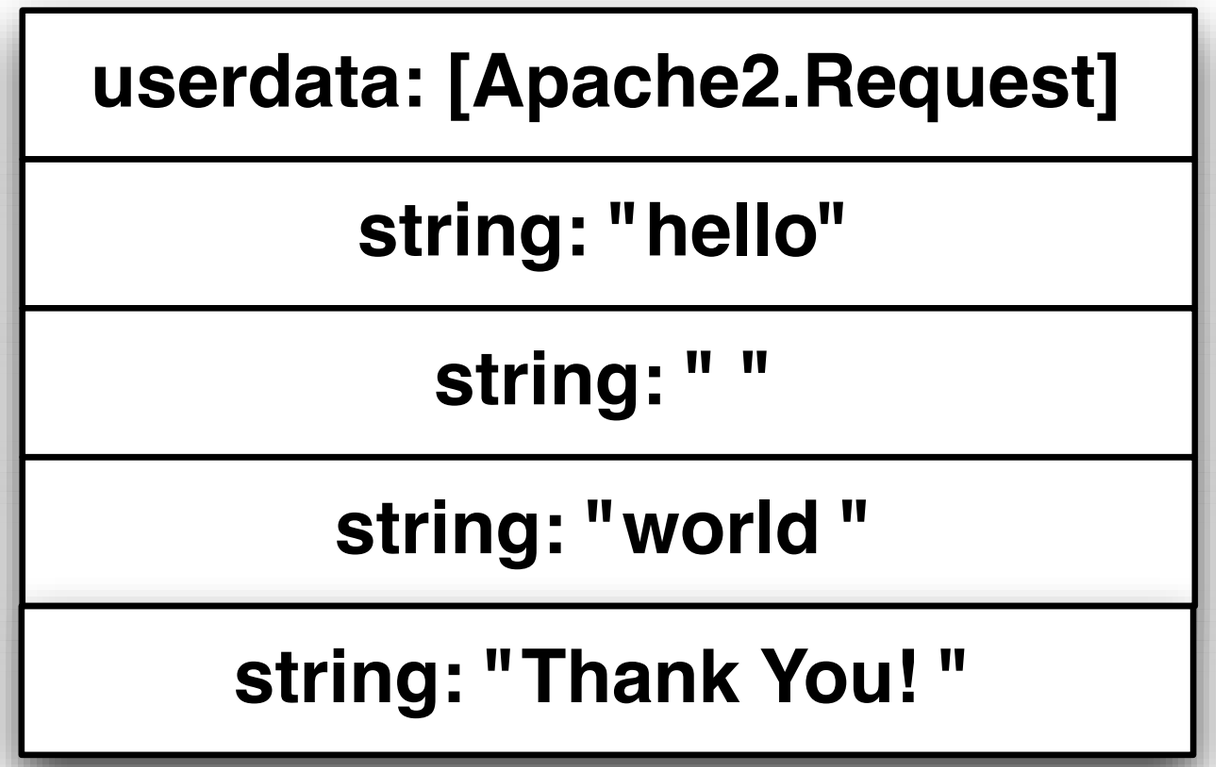
```
static int my_special_puts(lua_State* L) {
  request_rec* r = check_request_rec(L, 1);

  int argc = lua_gettop(L);
  int i;
  for (i=2;i<=argc;i++) {
    ap_rputs(luaL_checkstring(L, i), r);
  }
  lua_pushstring(L, "Thank You!");
  return 1;
}
```


userdata: [Apache2.Request]
string: "hello"
string: " "
string: "world "

```
static int my_special_puts(lua_State* L) {
  request_rec* r = check_request_rec(L, 1);

  int argc = lua_gettop(L);
  int i;
  for (i=2;i<=argc;i++) {
    ap_rputs(luaL_checkstring(L, i), r);
  }
  lua_pushstring(L, "Thank You!");
  return 1;
}
```

```
static int my_special_puts(lua_State* L) {
  request_rec* r = check_request_rec(L, 1);

  int argc = lua_gettop(L);
  int i;
  for (i=2;i<=argc;i++) {
    ap_rputs(luaL_checkstring(L, i), r);
  }
  lua_pushstring(L, "Thank You!");
  return 1;
}
```

string: "Thank You! "


userdata: [Apache2.Request]
string: "hello"
string: " "
string: "world "
string: "Thank You! "

```
function handle(r)
  local msg = r:myputs("Hello", " ", "world")
  r:debug(msg)
end
```

[Tue Nov 11 22:01:34 2007] [debug] @/www/ac_us_08/lib/cstuff.lua(3): [client 127.0.0.1] Thank You!

Evolving!

```
int luaopen_myputs(lua_State *L) {
  luaL_getmetatable(L, "Apache2.Request");
  lua_pushcfunction(L, my_special_puts);
  lua_setfield(L, -2, "myputs");
  lua_newtable(L);
  return 1;
}
```

Evolving!

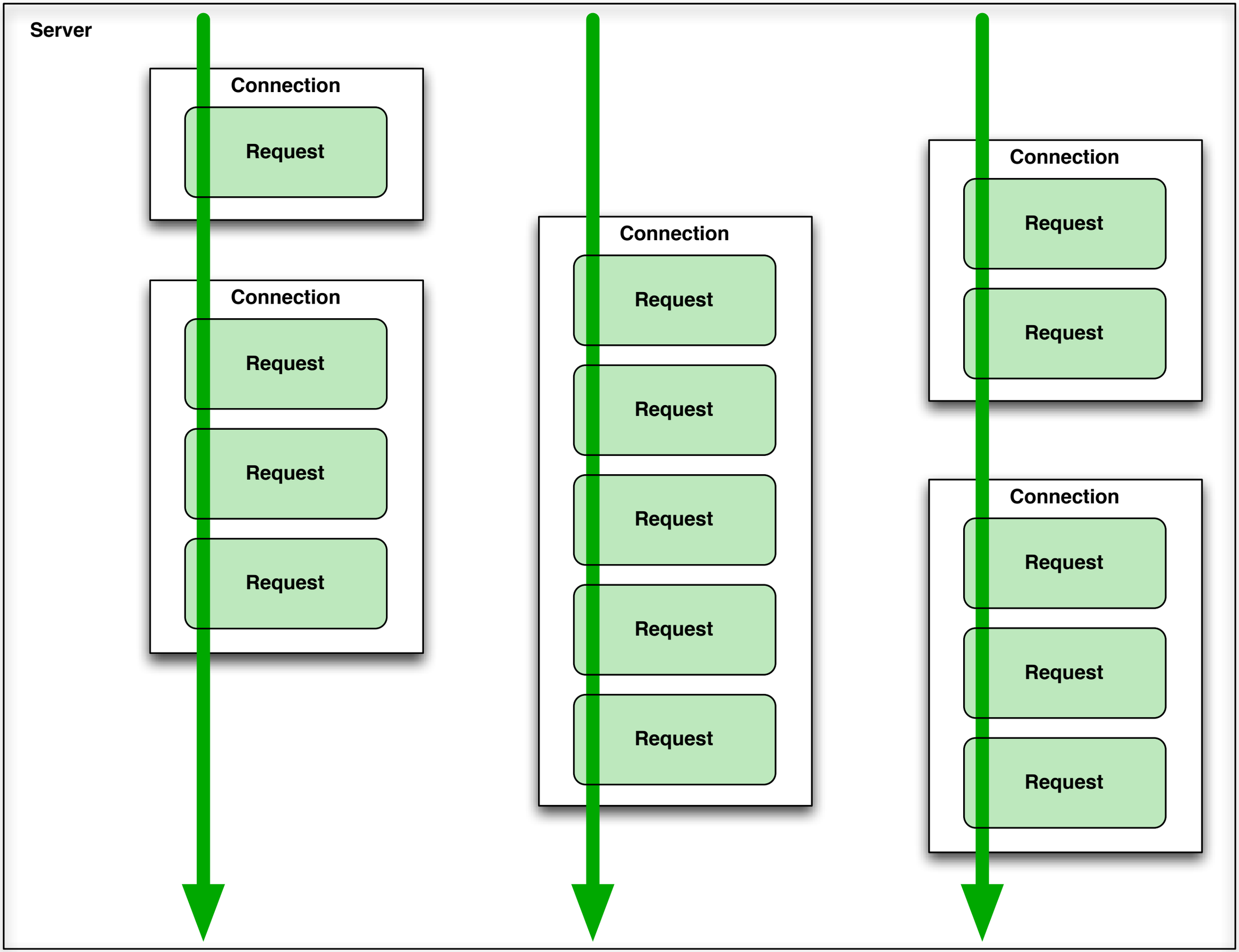
```
int luaopen_myputs(lua_State *L) {
    ...
}
```

Evolving!

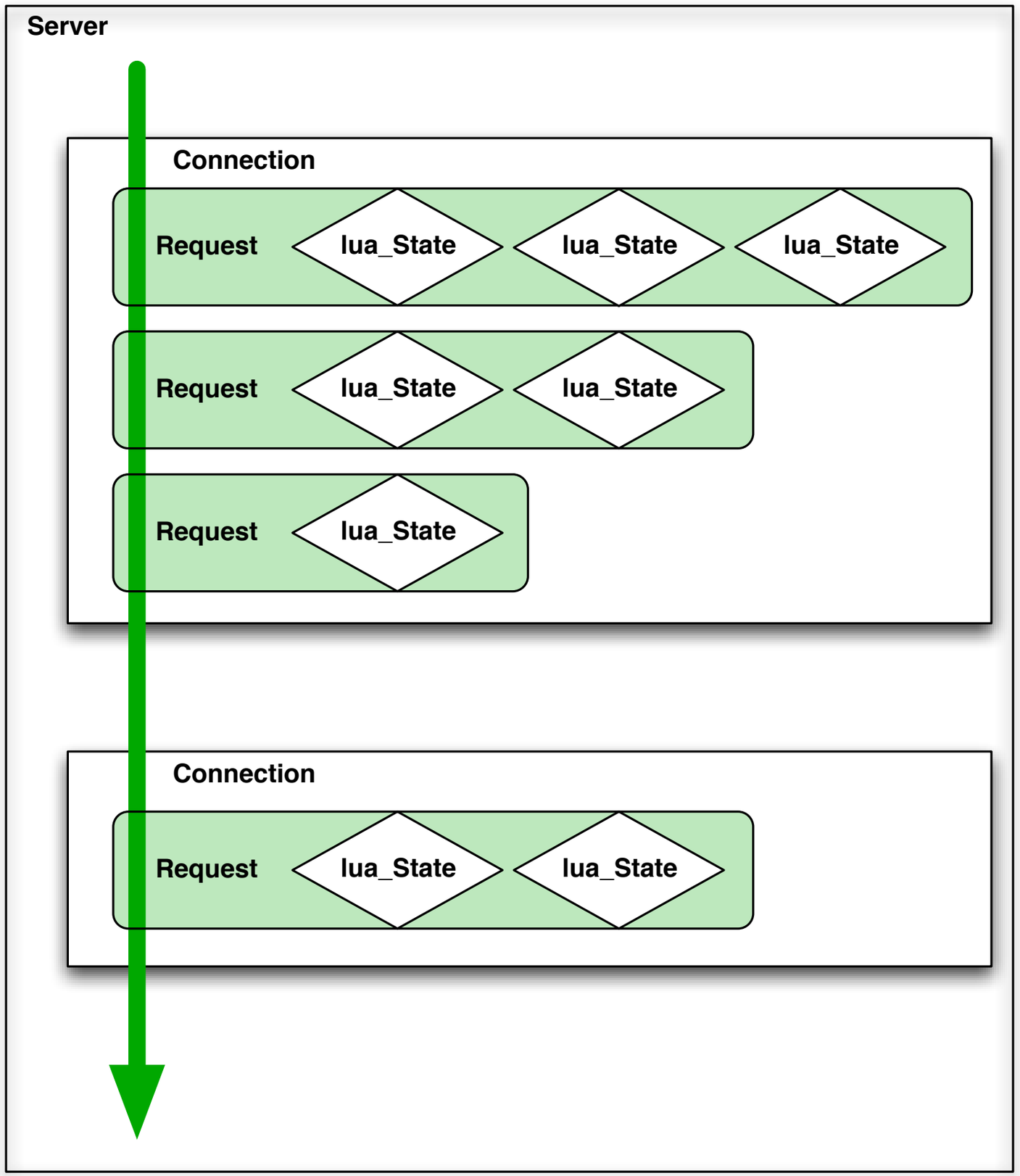
```
lua_getfield(L, LUA_REGISTRYINDEX,
             "Apache2.Request.dispatch");
apr_hash_t* dispatch = lua_touserdata(L, 1);
lua_getfield(L, LUA_REGISTRYINDEX,
              "Apache.Wombat.pool");
apr_pool_t* pool = lua_touserdata(L, 1);
apr_hash_set(dispatch,
             "myputs",
            APR_HASH_KEY_STRING
             apw_make_req_field(
                 &my_special_puts,
                 APW_REQ_FUNTYPE_LUACFUN,
                 pool
             );
```

How Wombat Works

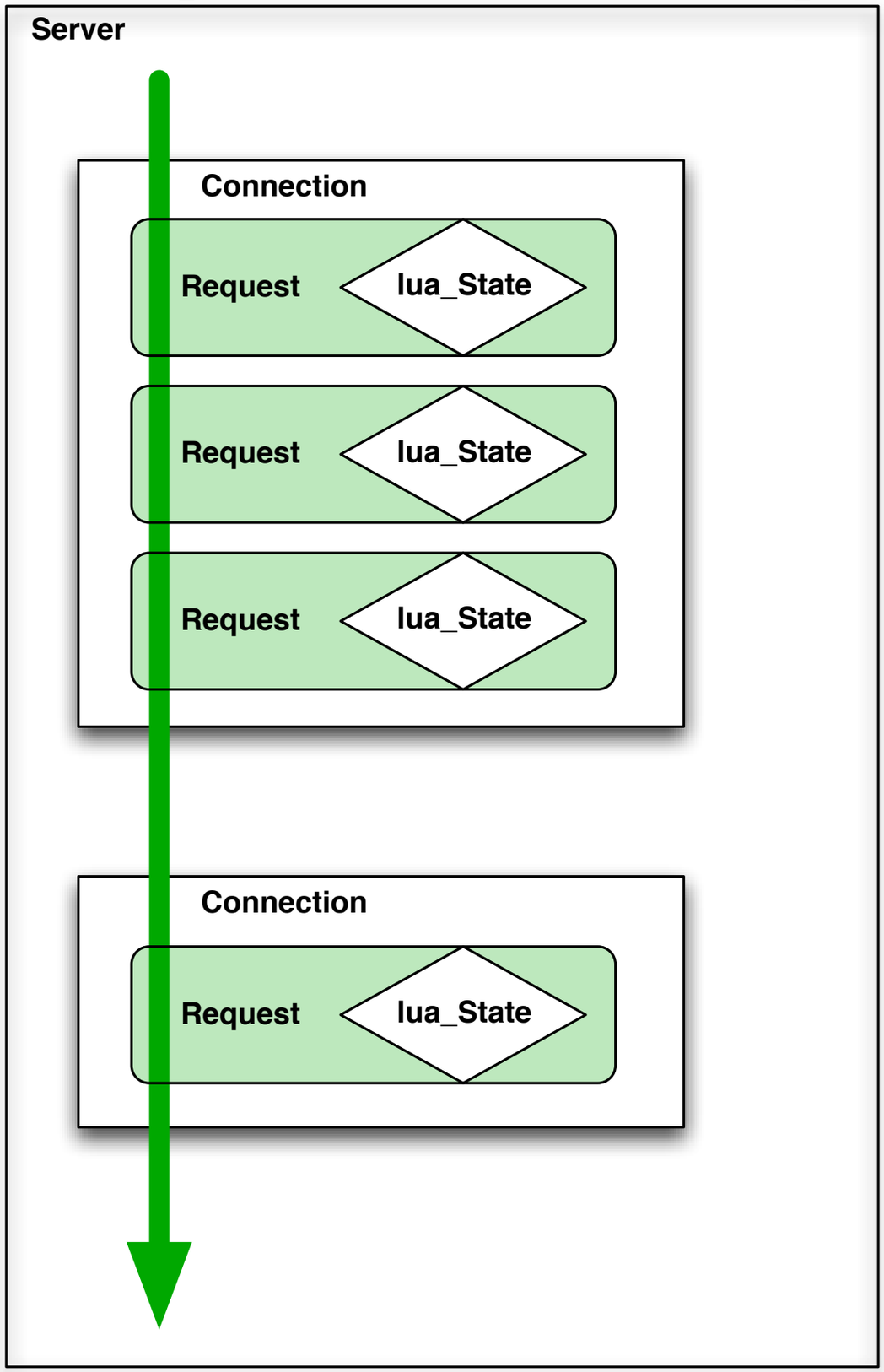
Differently then others


Server
Connection
Request
Request
Connection
Request
Request
Connection
Request
Request
Request
Request
Connection
Request
Request
Connection
Request
Request
Request

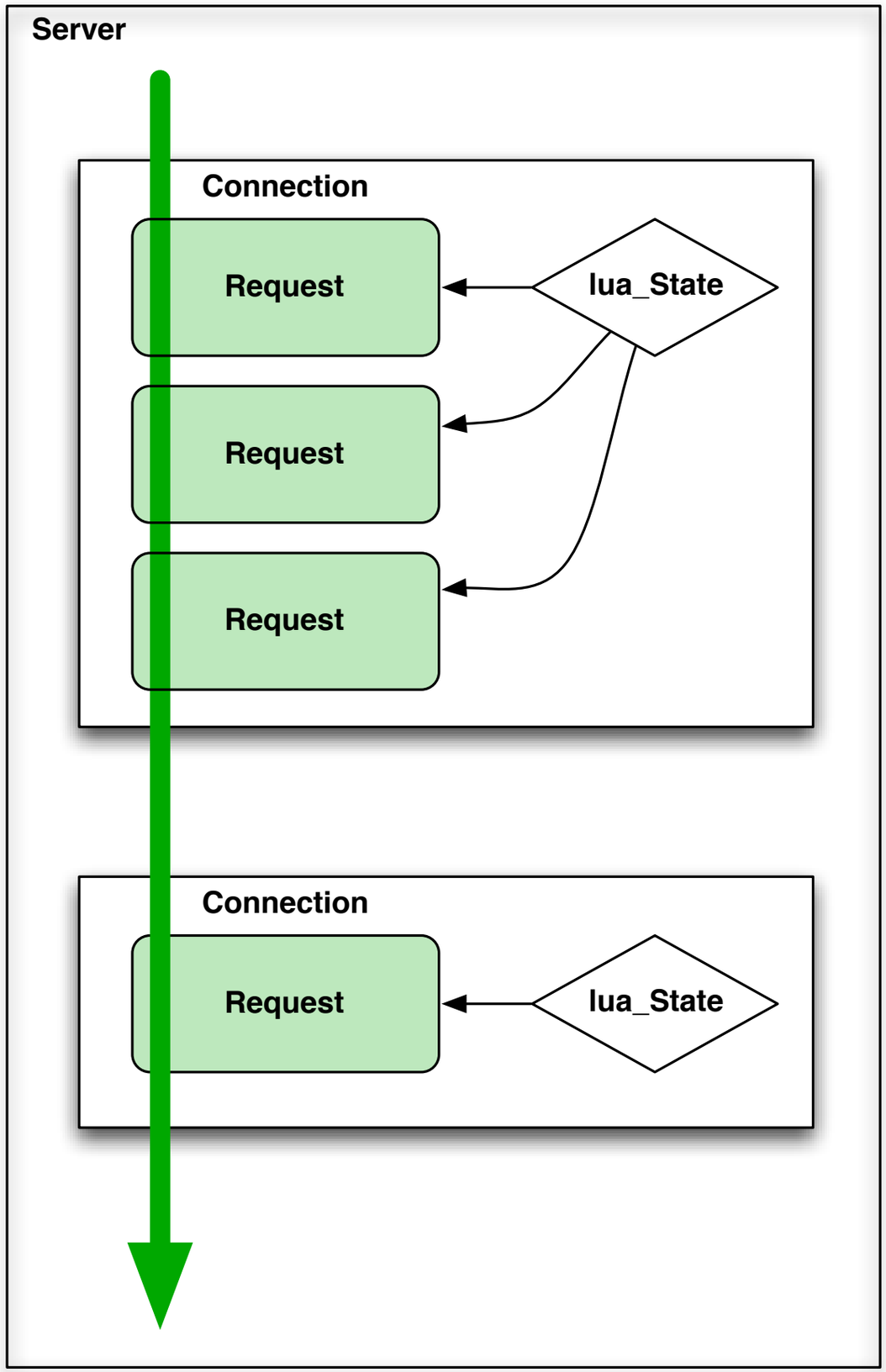
Once Scope


Server
Connection
Request
lua_State
lua_State
lua_State
Request
lua_State
lua_State
Request
lua_State
Connection
Request
lua_State
lua_State

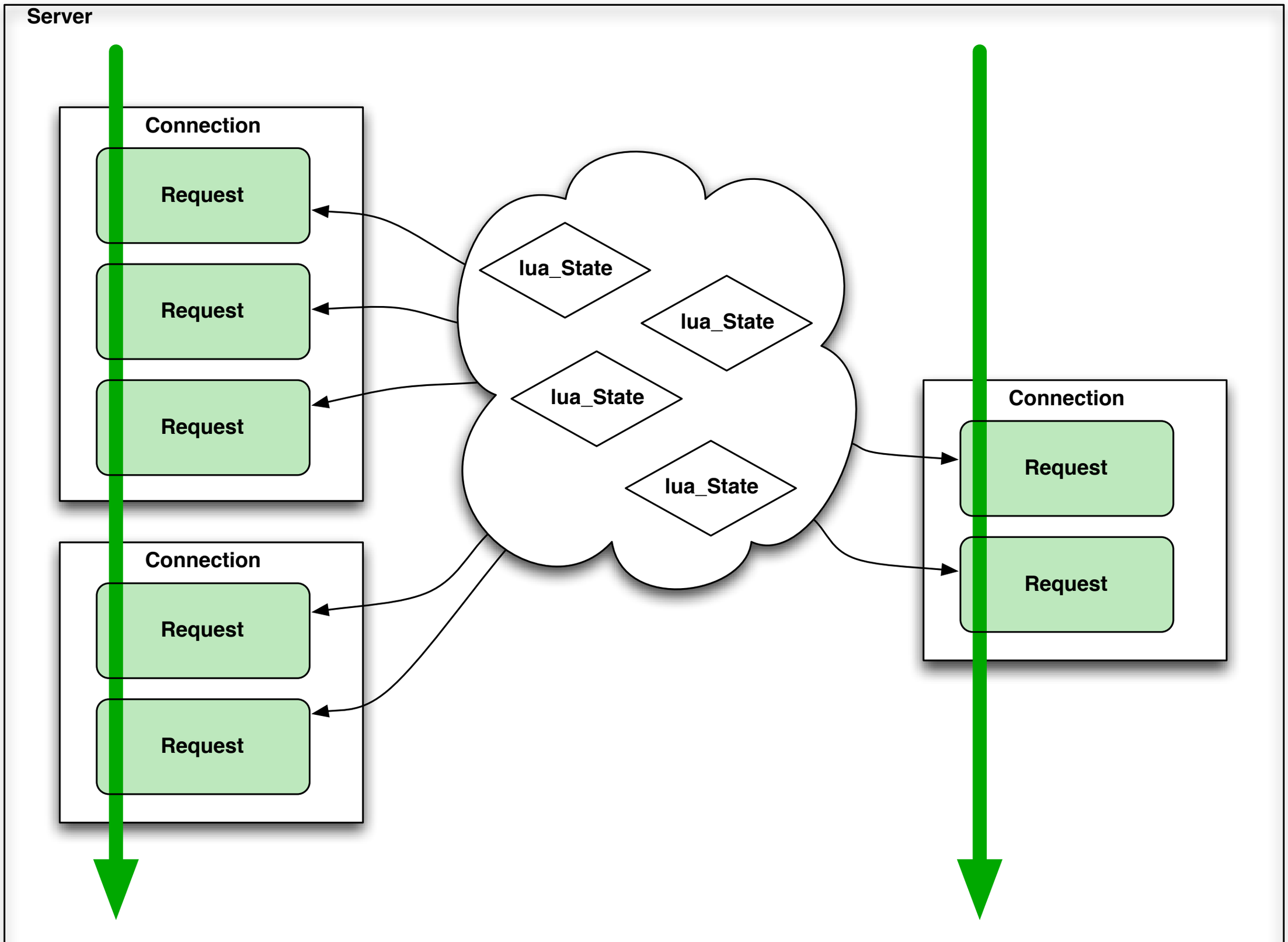
Request Scope

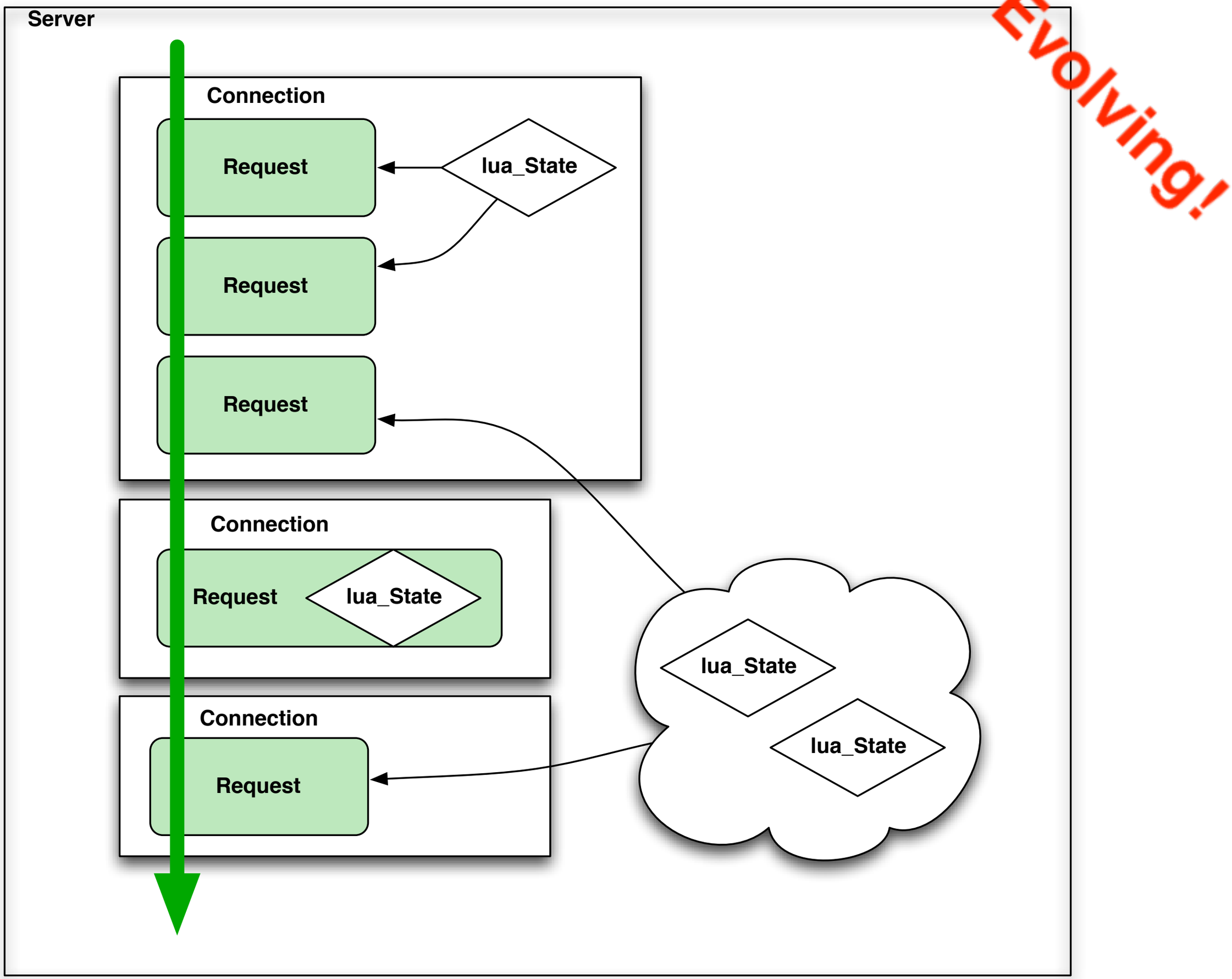


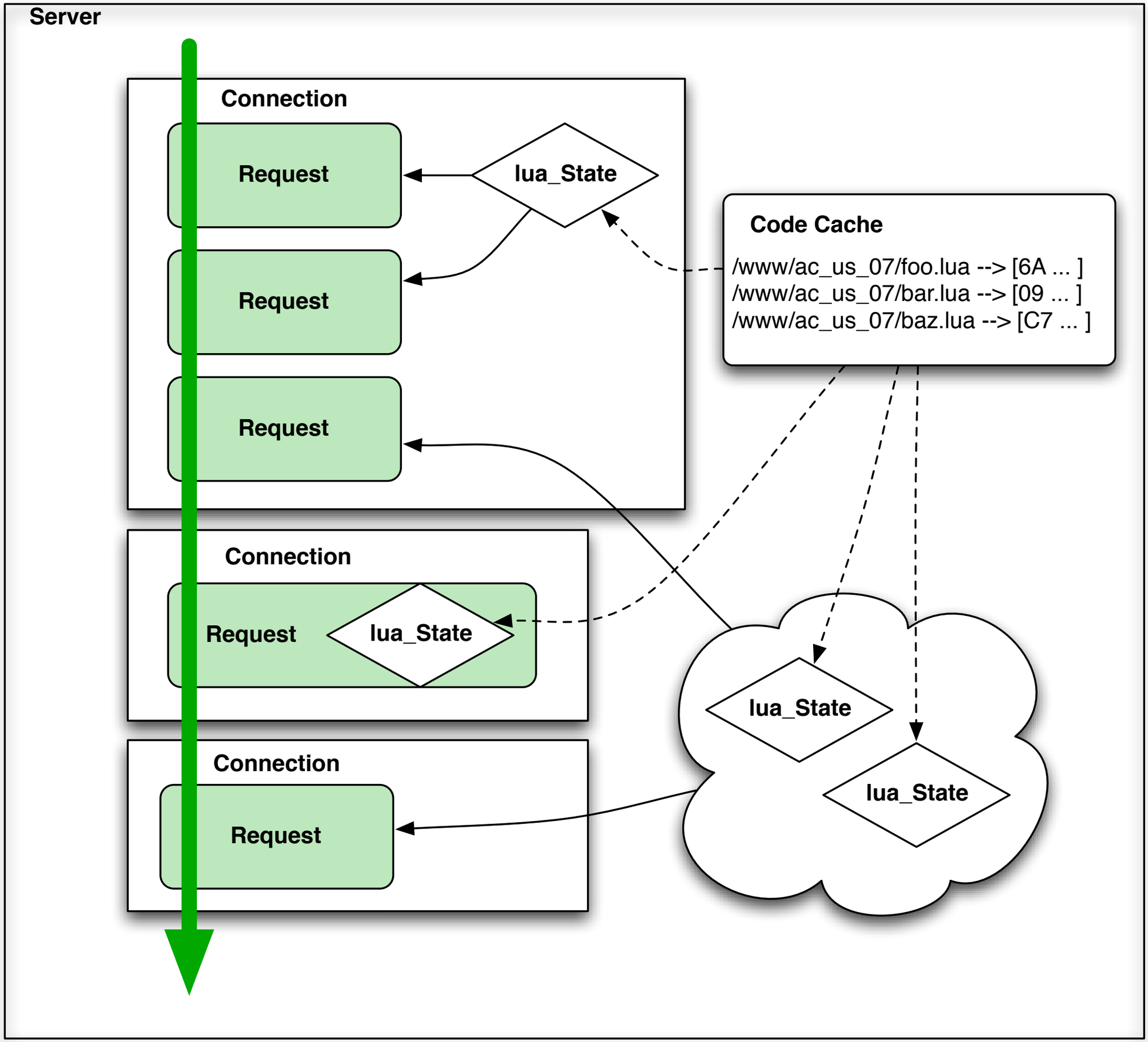
Connection Scope



Server Scope




Server
Connection
Request
lua_State
Request
Request
Connection
Request
lua_State
Connection
Request
lua_State
lua_State
Evolving!


Server
Connection
Request
lua_State
Request
Request
Code Cache
/www/ac_us_07/foo.lua --> [6A ...]
/www/ac_us_07/bar.lua --> [09 ...]
/www/ac_us_07/baz.lua --> [C7 ...]
Connection
Request
lua_State
Connection
Request
lua_State
lua_State

Evolving!

```
typedef struct {
    const char *file;
    int code_cache_style;
    int scope;

    // APW_SCOPE_ONCE only
    apr_pool_t *pool;

    apr_array_header_t* package_paths;
} apw_vm_spec;
```

Evolving!

```
lua_State *L = apw_rgetvm(r, spec);

lua_getglobal(L, d->function_name);

if (lua_pcall(L, 1, 0, 0)) {
    report_lua_error(L, r);
}
```

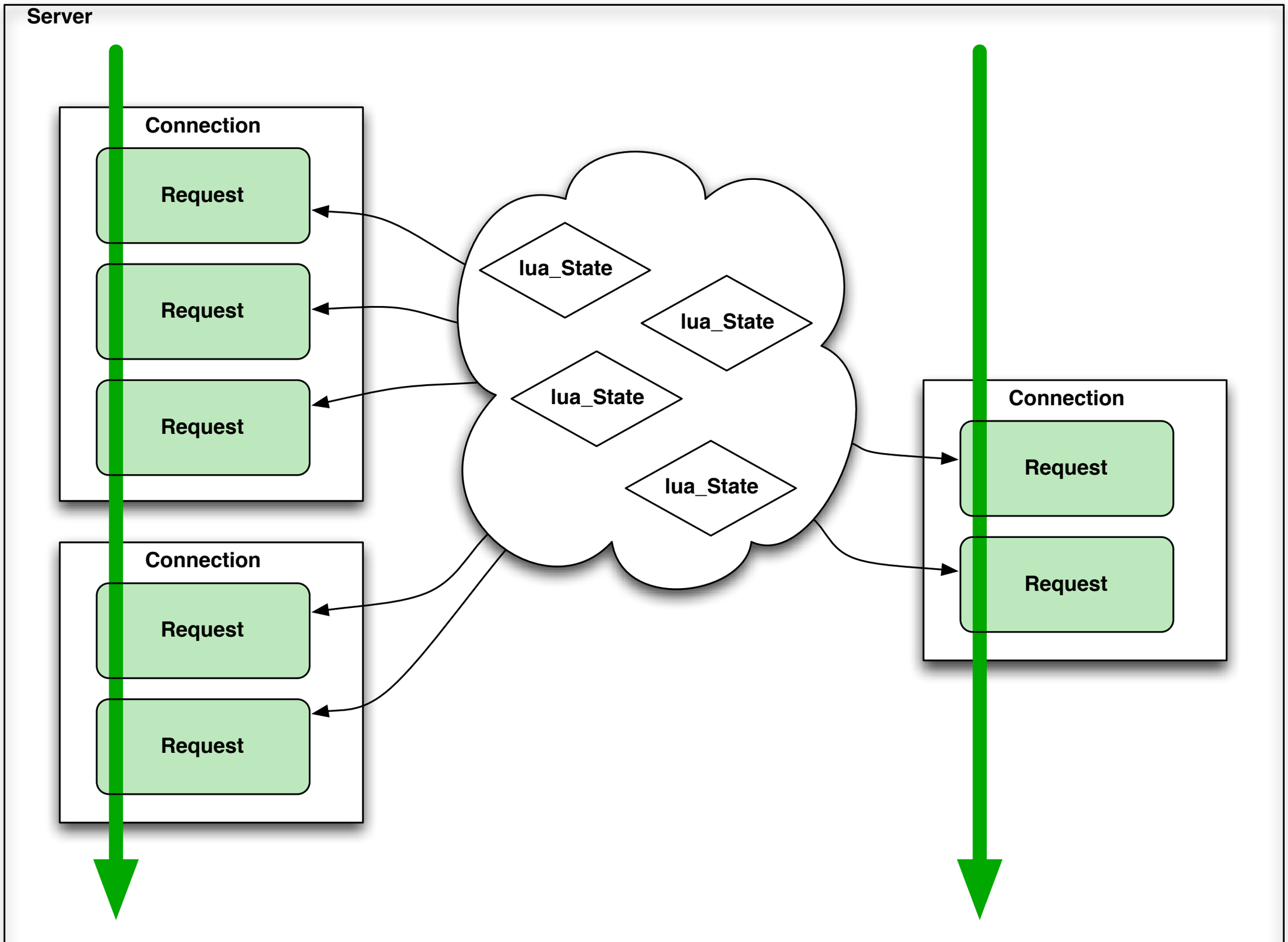
State (and Future) of the Wombat


TOOL

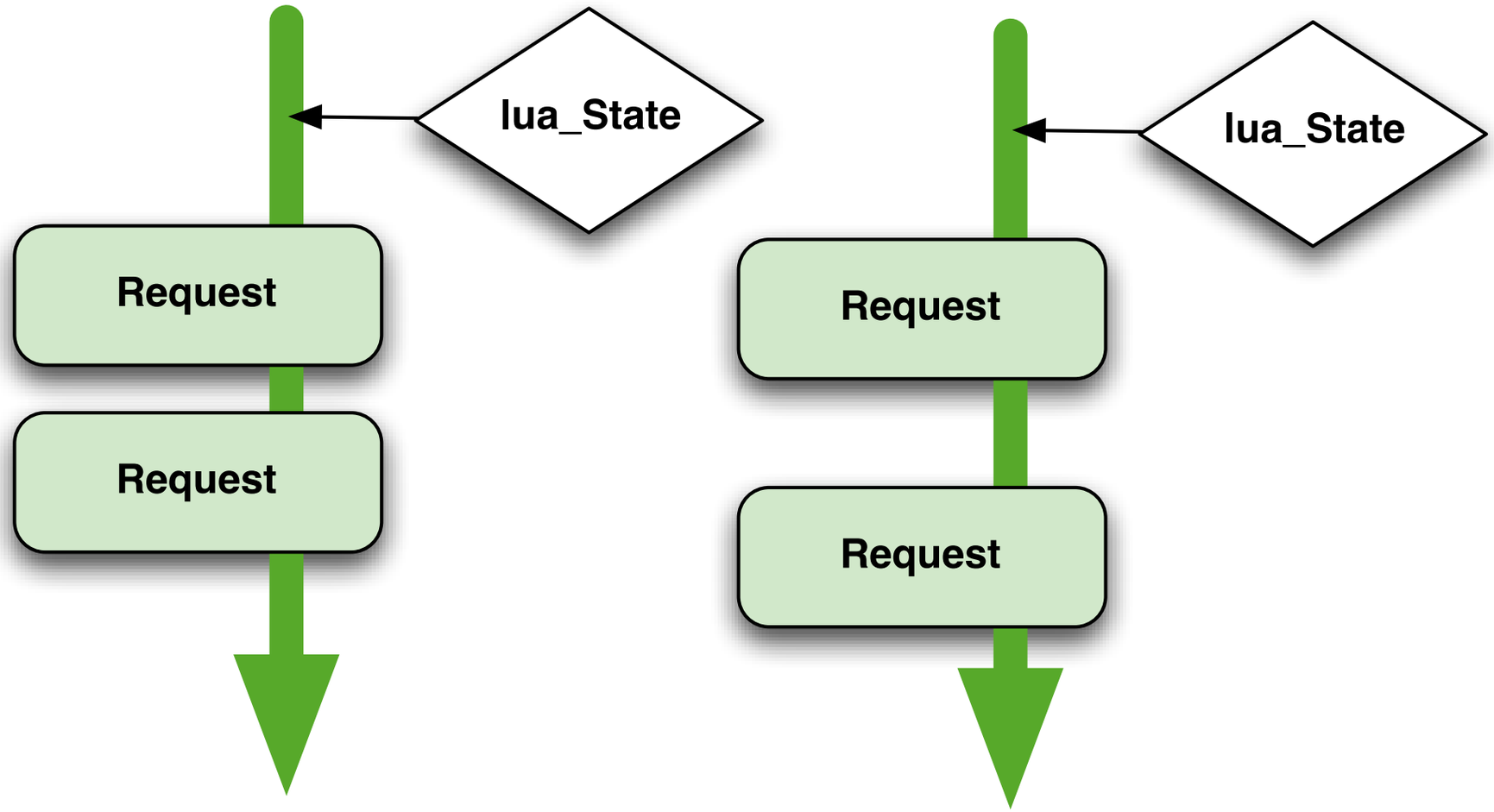
Evolving!

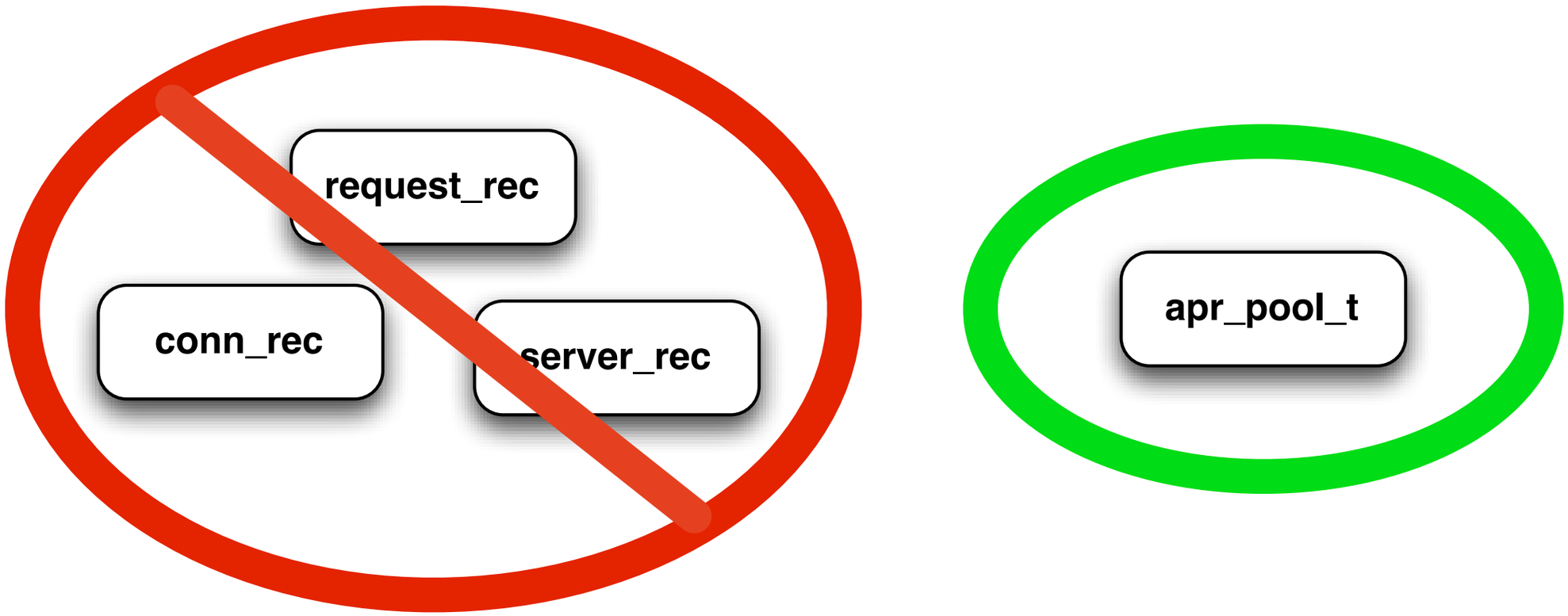
```
lua_State *L = apw_get_lua_state(
   pool,                  // apr_pool_t*
   file_path,             // char*
   package_paths,   // apr_array_header_t
   package_cpaths,  // apr_array_header_t
   &wombat_open_callback, baton);
```

Server Scope



Thread Scope




request_rec
conn_rec
server_rec
apr_pool_t

The URL

http://svn.apache.org/repos/asf/httpd/mod_wombat

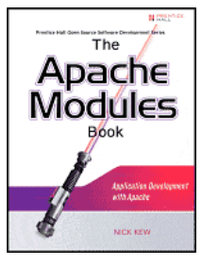
Resources & Docs

http://www.lua.org/

http://www.lua.org/manual/5.1/

http://www.lua.org/pil/

Programming in Lua, Roberto Ierusalimschy



The Apache Modules Book, Nick Kew

That's All Folks!

Brian McCallister

http://kasparov.skife.org/wombat _ac_us_08.pdf

Bonus!

Because we have spare time...

```
local mu = require "moonunit"
local http = require "helpers"

http.base_url = "http://localhost:8000"

local test = mu.TestCase:new{}

function test:basic_get()
  local body, code = http.get "/basic"
  assert(200 == code,
         "expected 200, got " .. code)
  assert(body:find("hello Lua world"))
end

test:run()
```

Terminal — bash — bash — 108×29 — ⌘2

```
Last login: Tue Nov 13 21:44:15 on ttys002
brianm@85:~$ cd src/wombat/test/
brianm@85:~/src/wombat/test$ ./test.lua
[basic_post_alt]                        pass
[map_regex2]                            pass
[server_says_hi]                        pass
[post_with_table]                       FAIL ./test.lua:51: expected status code 200, got 404
[basic_post]                            FAIL ./test.lua:38: expected status code 200, got 404
[request_attributes]                    pass
[simple]                                pass
[map_regex]                             pass
[translate_name_hook2]                  pass
[fixups_hook]                           pass
[server_version]                        pass
[request_scope_says_hi]                 pass
[basic_get]                             FAIL ./test.lua:26: expected status code 200, got 404
[super_basic_config]                    pass
[simple_mapped]                         pass
[quietly]                               pass
[translate_name_hook]                   pass
[simple_filter]                         pass
brianm@85:~/src/wombat/test$
brianm@85:~/src/wombat/test$
brianm@85:~/src/wombat/test$ ./test.lua simple_filter basic_post
[simple_filter]                         pass
[basic_post]                            FAIL ./test.lua:38: expected status code 200, got 404
brianm@85:~/src/wombat/test$
```

That's (Really) All Folks!

Brian McCallister

http://kasparov.skife.org/wombat _ac_us_08.pdf

(cc) <smeltzy> danelle http://flickr.com/photos/danellemac/1704989073/