mod_wombat

Apache + Lua + Worker For The Win!

Brian McCallister
Ning

```
require 'apache2'
require 'math'

s = "/www/ac_us_07/htdocs/%s.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
```

I got really tired of cases where apache comes >< close to exactly what I want. mod_rewrite not enough, mod_perl too... heavy and perlish, mod_ruby... no threads, mod_python, GiL and HUGE memory footprint. Need a really small, actual language, which you can do little, fast, efficient modules in.

Why Lua?

The obvious question from there is why did we use Lua?

Threading

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

The first thing is its thread behavior. It doesn't do threads, but it also encapsulates all interpreter state in a struct, so you can have as many as you want being accessed from different threads. Actually, it does to incredibly efficient cooperative multithreading, but not POSIX threads.

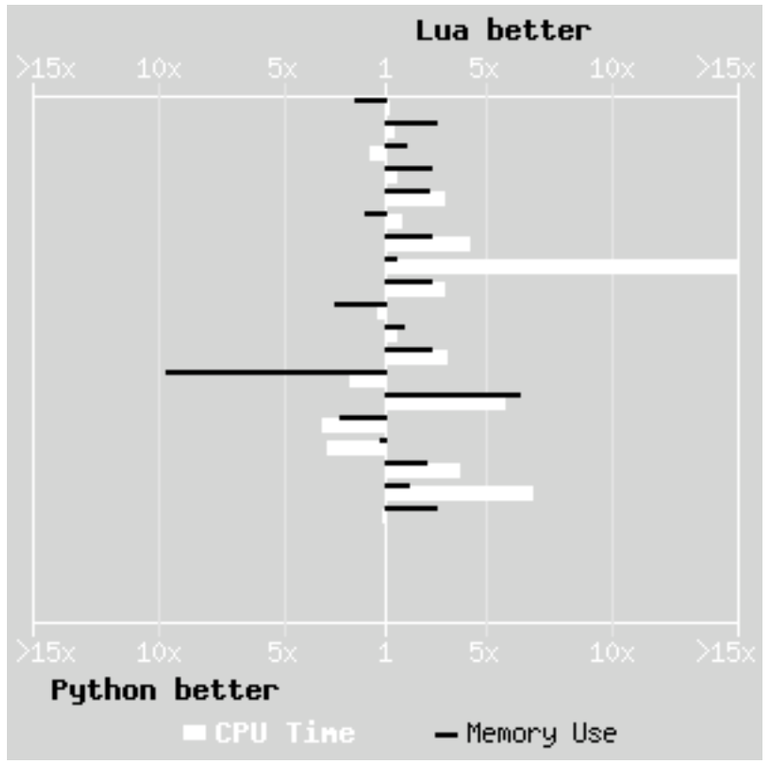
Overhead

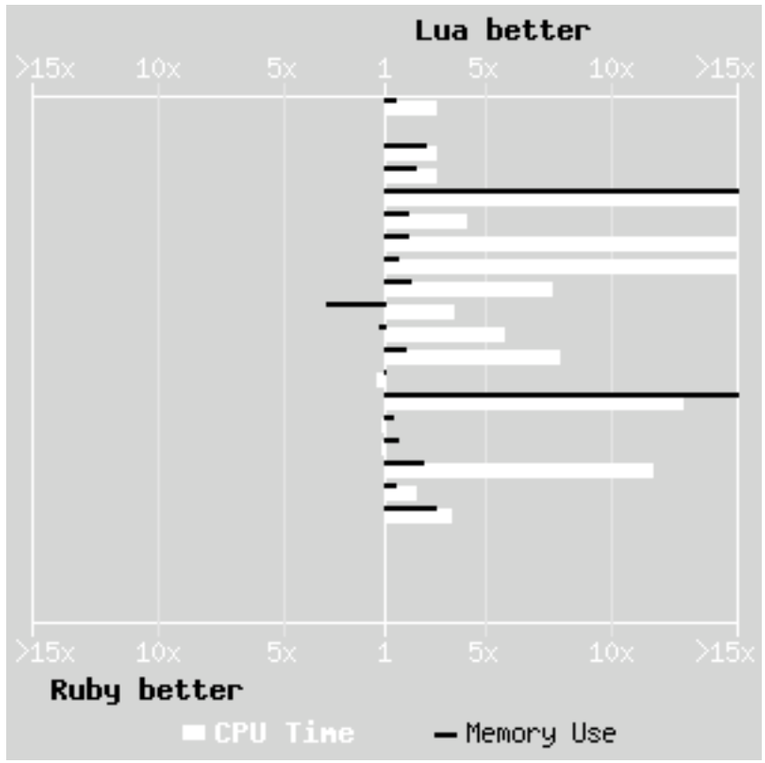
Server	Reqs / Second	Notes
Apache HTTPD mod_hello_world	18,823.58	minimal module which just prints hello world
Apache HTTPD mod_wombat	17,856.76	server-scoped mod_wombat handler
Apache Tomcat 5.5.20	17,644.40	JSP
Jetty 6.1.1	12,449.36	JSP
Mongrel	2,378.05	HttpHandler, not Rails

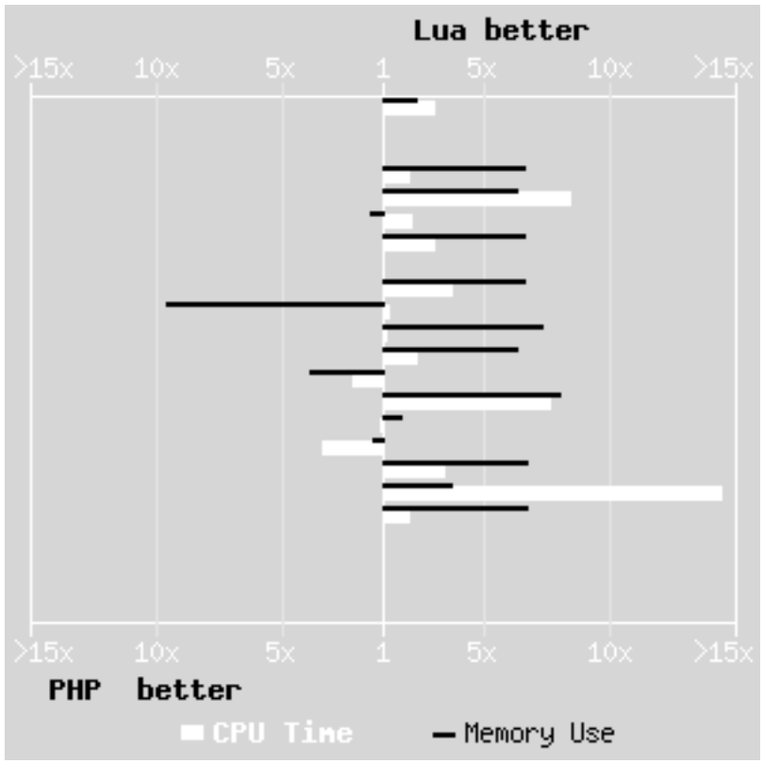
http://kasparov.skife.org/blog/src/ruby/silly-micro-benchmarks.html

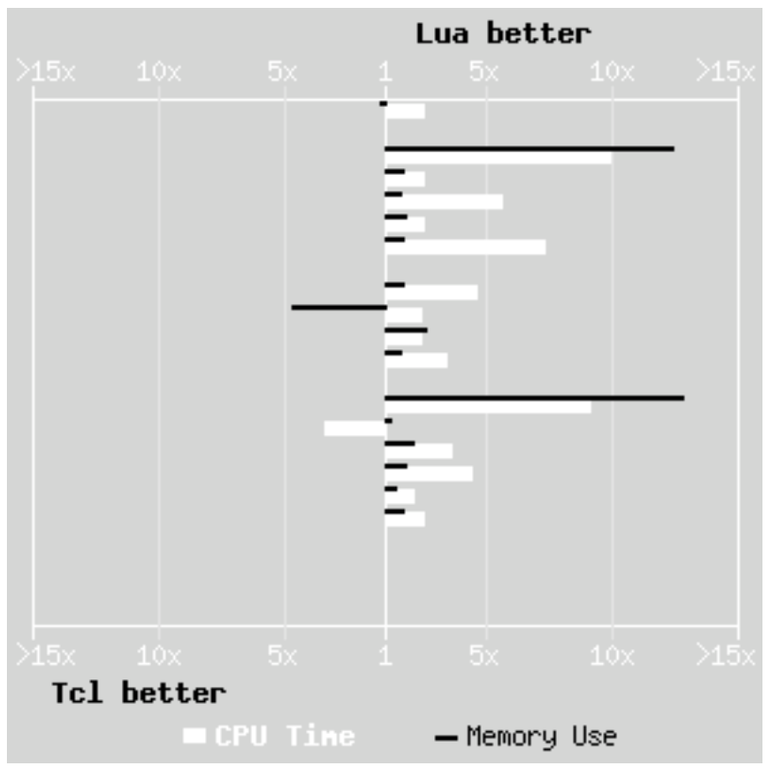
This invalid and pointless benchmark attempted to measure the cost of using a particular system. It did nothing but respond with “hello world” It is designed to see what the relative overhead to a baseline (mod_hello_world) is, mod_wombat did pretty nicely! In truth, the benchmark is even less valid. I had to run it twice before mod_hello_world was faster than mod_wombat.
15k or 16k RSS overhead per lua_State

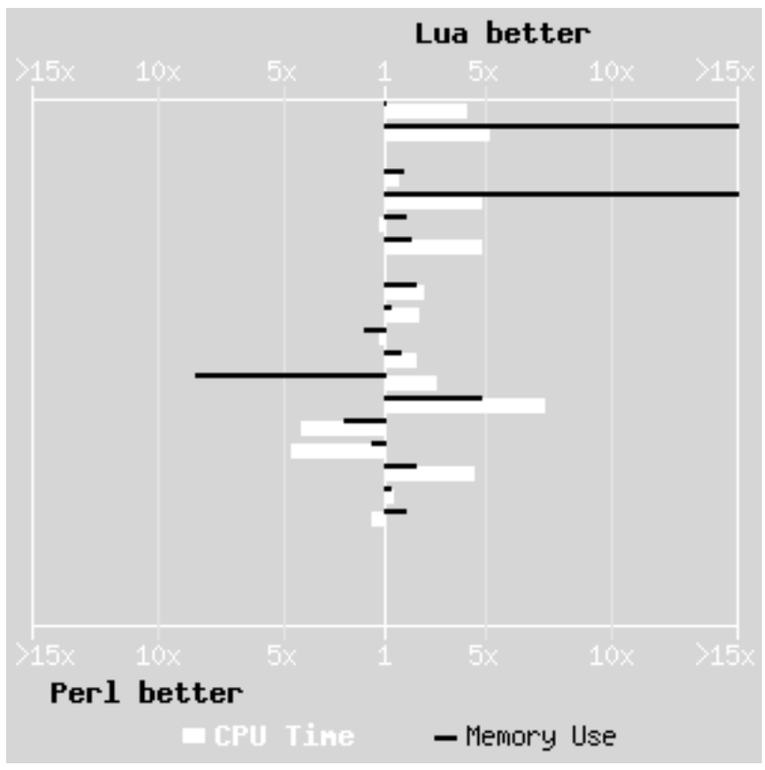
Performance

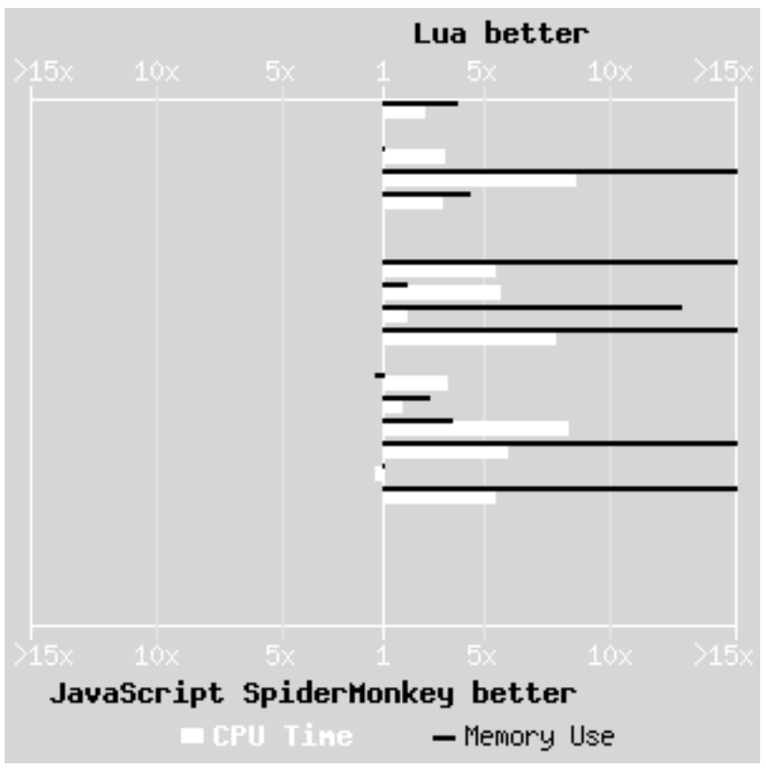












http://shootout.alioth.debian.org/gp4/benchmark.php

Third, Lua itself is actually damned fast. Not C fast, but it thrashes most other scripting languages in most cirumstances.

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;
}
```

Finally, it has to actually embed and play easily with C. Lua is 100% designed for... embedding in C/C++

Why Not $LANG ?

Guile

Gauche

Chicken

MZScheme

Elk

Scheme48

Python

SpiderMonkey

Perl

Ruby

TCL

PHP

Make note that Lua uses the MIT license, matters to people here :-)
* Because Vadim asked

Global Interpreter

Global Namespace

Thread Behavior

Size of Runtime

Line Noise :-)

Putting Wombat to Work

```lua
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")
```

Brief tour of Lua -- SMALL -- SIMPLE --
Tables Tables TABLES
dynamic typing, first class functions, tables, “OO” sugar on tables

```
local mu = require "moonunit"
local test = mu.TestCase:new{}
```

```
module("moonunit", package.seeall)

TestCase = {}
function TestCase:new(it)
  it = it or {}
  setmetatable(it, self)
  self.__index = self
  return it
end
```

```
ServerRoot "/opt/httpd-2.2.6"
DocumentRoot "/www/ac_us_07/htdocs"
Listen 80

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

LuaRoot "/www/ac_us_07/"

DirectoryIndex index.lua

AddHandler lua-script .lua

LuaHookTranslateName lib/hooks.lua \
                     rewrite_index
```

Point out the Lua bits, and the (current) dependency on apreq2

```
ServerRoot "/opt/httpd-2.2.6"
DocumentRoot "/www/ac_us_07/htdocs"
Listen 80

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

LuaRoot "/www/ac_us_07/"

DirectoryIndex index.lua

AddHandler lua-script .lua

LuaHookTranslateName lib/hooks.lua \
                     rewrite_index
```

The file is LuaRoot .. first arg to the directive
The function is second arg to the directive

```
-- /www/ac_us_07/lib/hooks.lua
require 'string'
require 'apache2'

htdocs = "/www/ac_us_07/htdocs

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = htdocs .. "/index.lua"
    return apache2.OK
  end
  return apache2.DECLINED
end
```

our translate_name hook function

```
-- /www/ac_us_07/lib/hooks.lua
require 'string'
require 'apache2'

htdocs = "/www/ac_us_07/htdocs

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = htdocs .. "/index.lua"
    return apache2.OK
  end
  return apache2.DECLINED
end
```

First, note the file (first line)
and note the function name
NEXT SLIDE

```lua
-- /www/ac_us_07/lib/hooks.lua
require 'string'
require 'apache2'

htdocs = "/www/ac_us_07/htdocs

function rewrite_index(r)
  if r.uri == "/index.html" then
    r.filename = htdocs .. "/index.lua"
    return apache2.OK
  end
  return apache2.DECLINED
end
```

Look at function itself. The argument is the request_rec (sorta).
Access the standard fields on it just as lua table entries, both read and write (uses scary metatable hackery)
apache2.OK and apache2.DECLINED are the same as the C constants
(Should map in the DocumentRoot, LuaRoot, etc somewhere)

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

Just to point out, this is just setting and reading from the request_rec

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 because the dependency on libapreq2 will probably go away. Exactly how the API will be structured then is unknown. I like this way a lot, so it may just be that we'll test for apreq's availability and do it with apreq if avail, and parse out in Lua if not avail, but API will be the same. One will just be more efficient and leave the args around for C modules

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!

```
-- /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
```

returning tuples, ir the parseargs()
traversing array
libapreq2 is becoming optional, may change to something like: apreq.parsearegs(r)

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
```

returning tuples, ir the parseargs()
traversing array
libapreq2 is becoming optional, may change to something like: apreq.parsearegs(r)

```
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
```

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

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

Evolving -- things other than request_rec are fuzzy as changing how we map

```
LuaRoot "/www/ac_us_07/"


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
```

Evolving!

Names are likely to be normalized Real Soon Now
TODO: Get ordering here to match order of execution

On the C Side

(Briefly)

```
require "myputs"

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

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;
}
```

```
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! "

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

[Tue Nov 11 22:01:34 2007] [debug] @/www/ac_us_07/lib/cstuff.lua(3): [client 127.0.0.1] Thank You!

```
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;
}
```

this is a BAD BAD BAD module

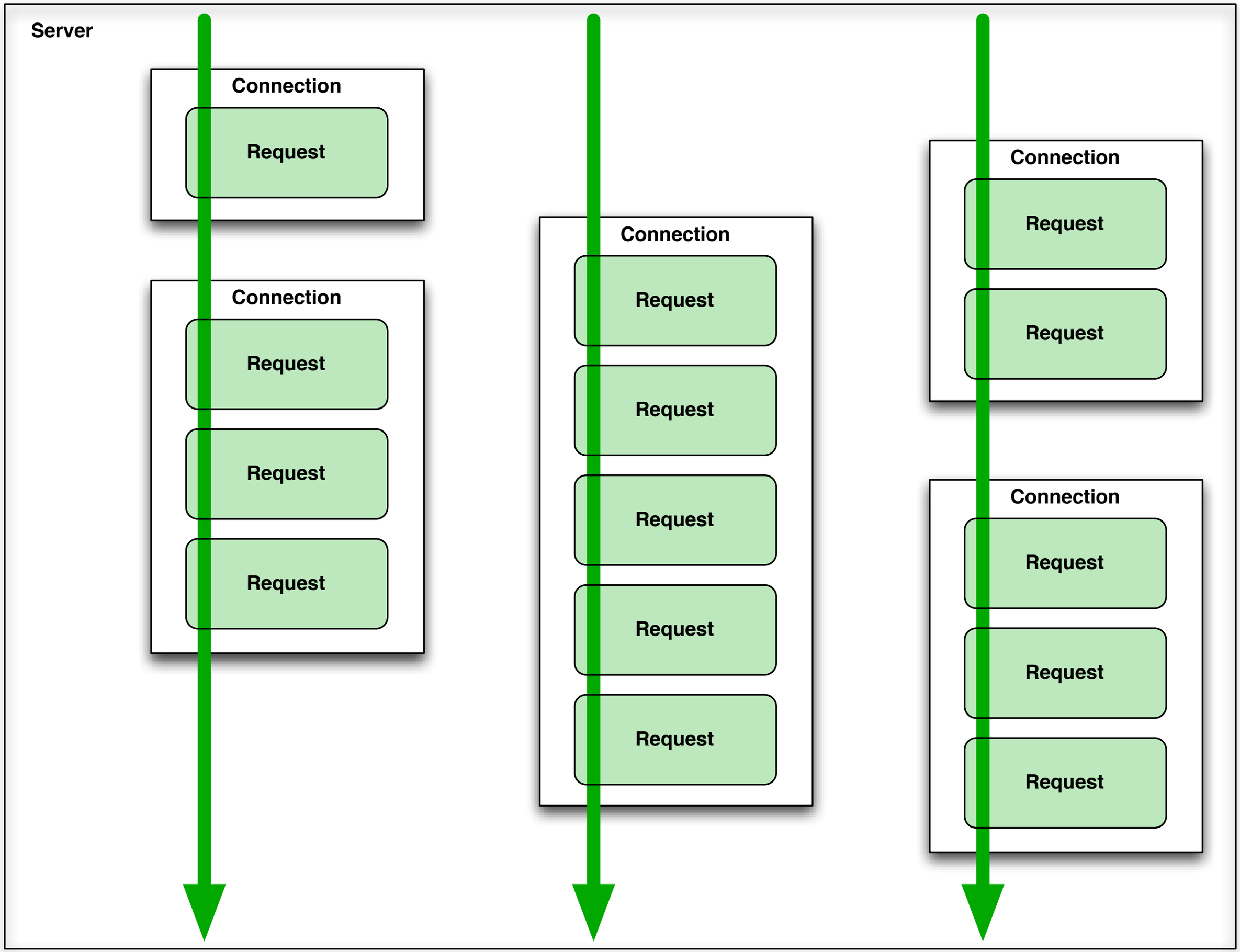
```
int luaopen_myputs(lua_State *L) {
    luaL_getmetatable(L, "Apache2.Request");
    lua_pushcfunction(L, my_special_puts);
    lua_setfield(L, -2, "myputs");
    lua_newtable(L);
    return 1;
}
```

this is a BAD BAD BAD module

How Wombat Works

Differently then others

As mentioned, one of the driving forces behind mod_wombat is working well with Apache's threading model.

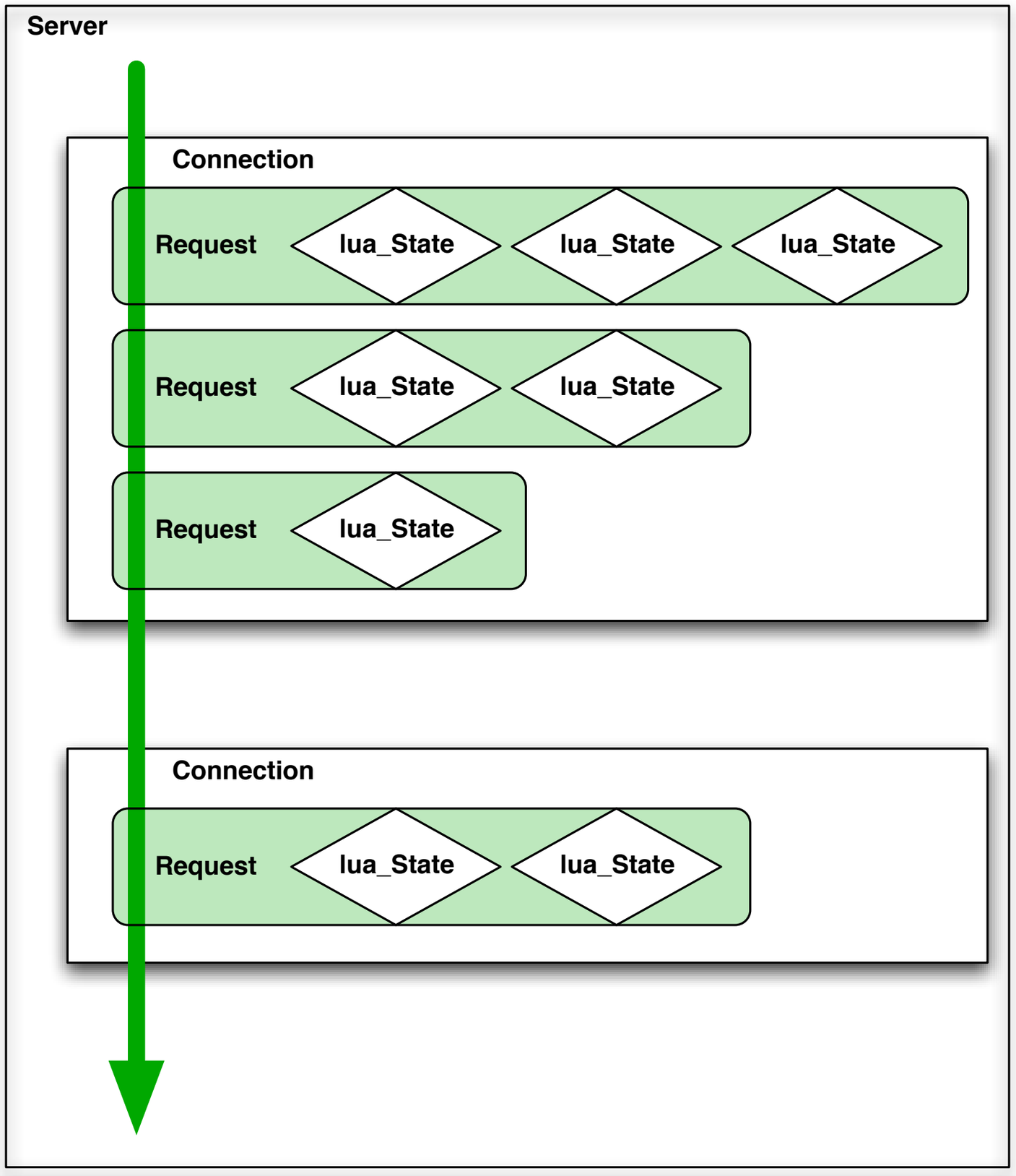


Two things to review before diving in. The first is how the Worker MPM works...

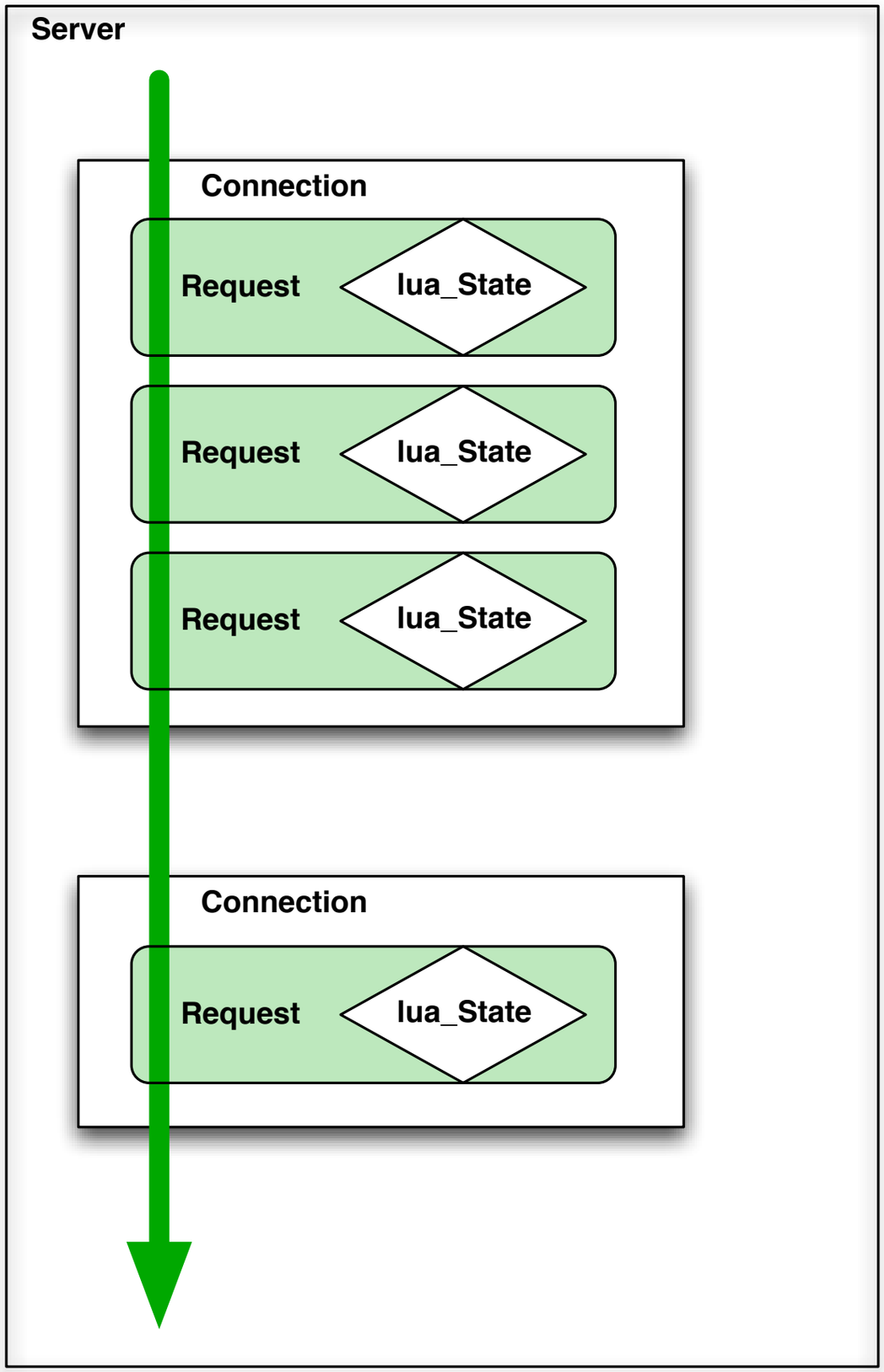
```
lua_State *L = apw_rgetvm(r, spec);

lua_getglobal(L, d->function_name);

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

The second is using a lua_State (interpreter struct)

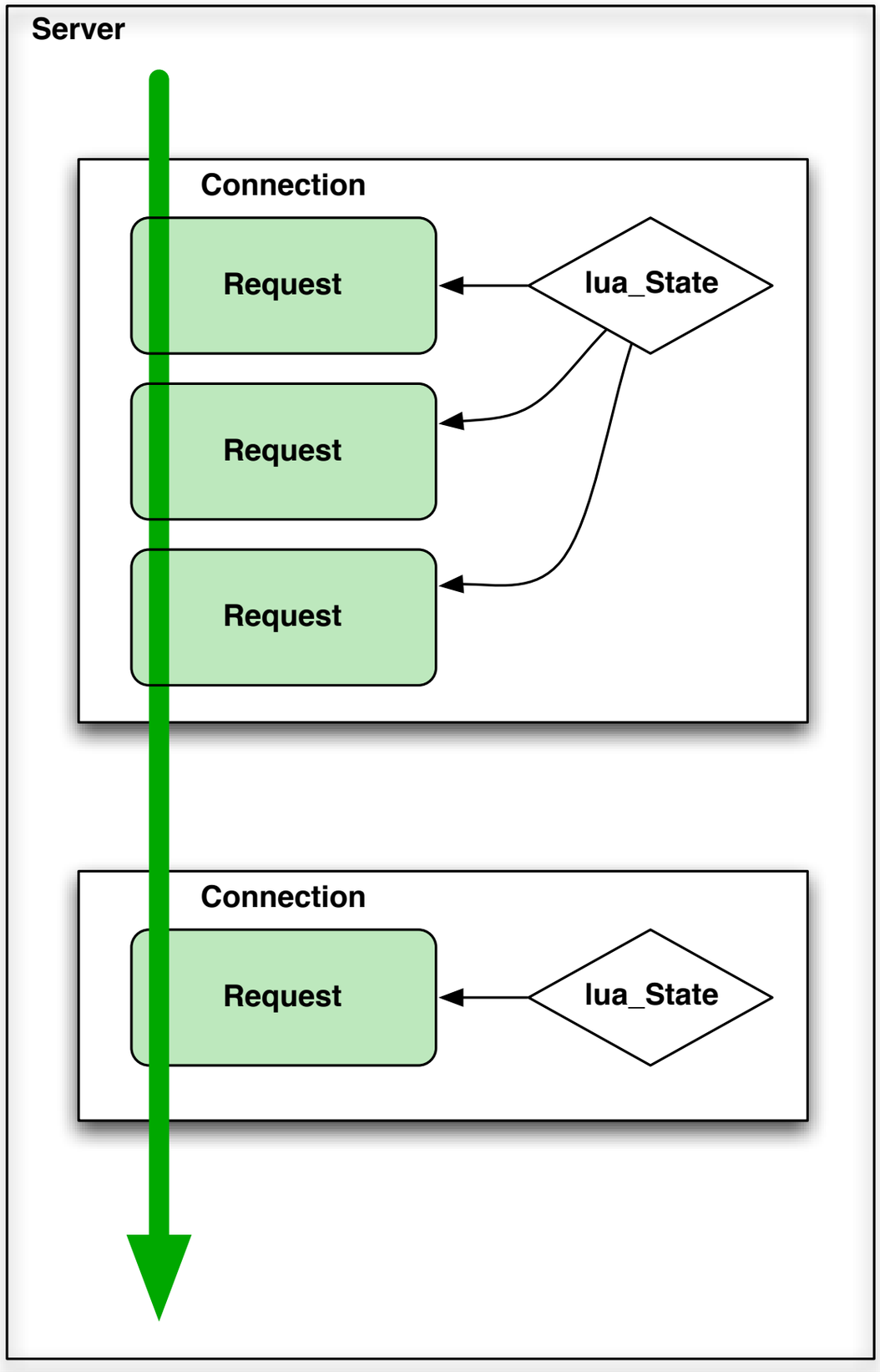
Once Scope


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

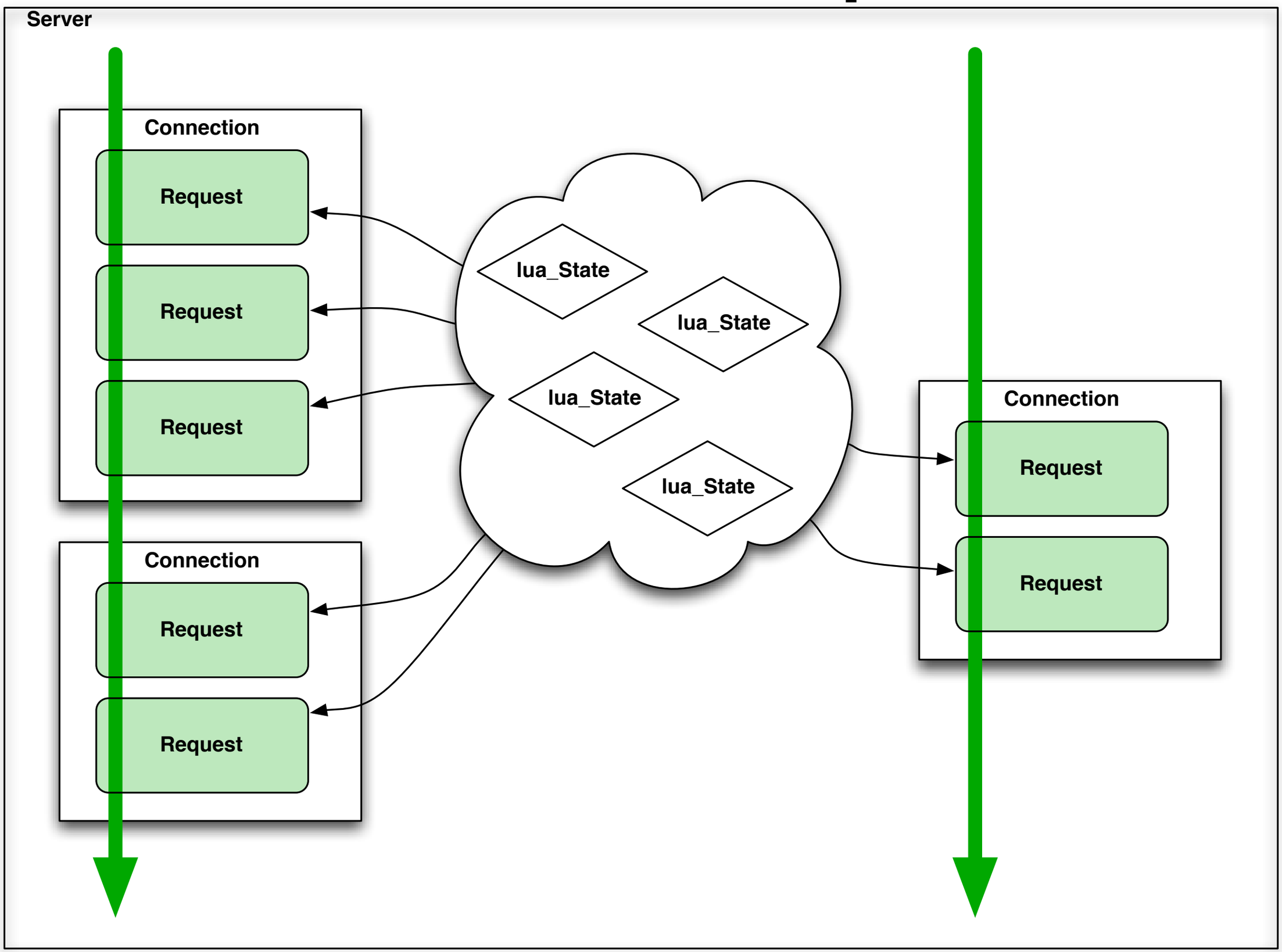
Request Scope

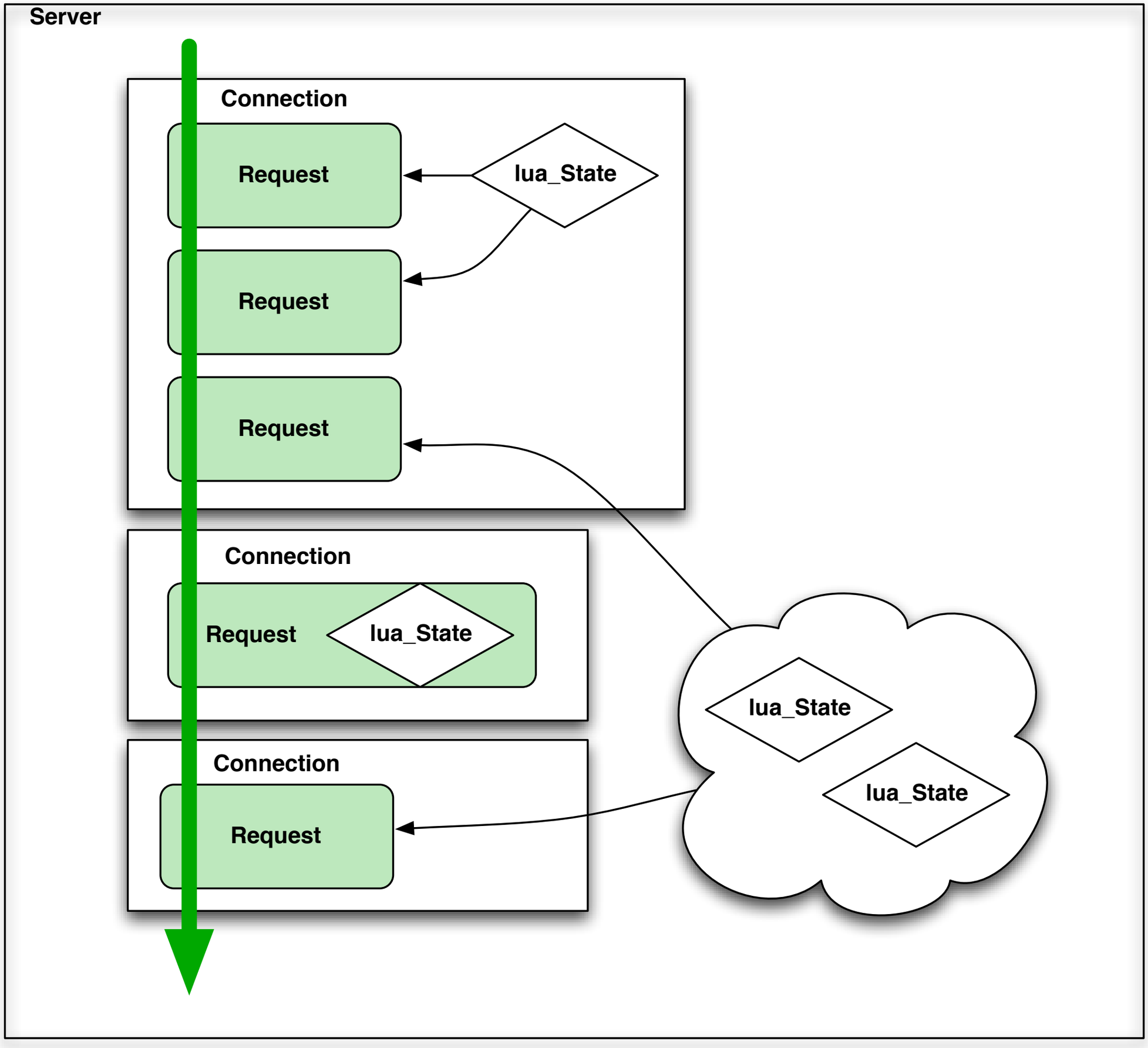


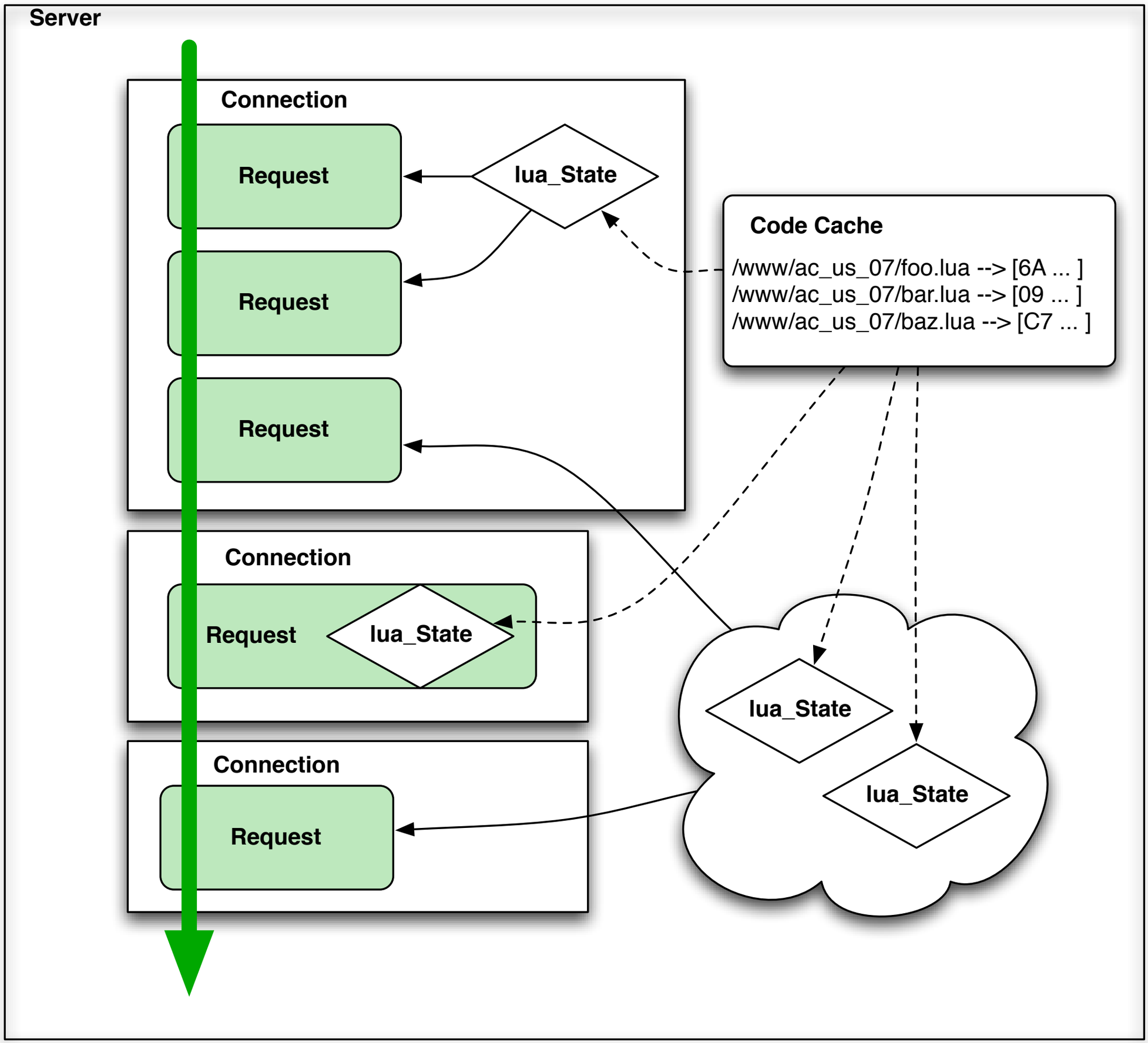
Connection Scope



Server Scope




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



NEVER, STAT, FOREVER
Once and Stat are the defaults

```
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;
```

```
LuaRoot "/www/ac_us_07/"

LuaConfig httpd_config.lua configure
```

Evolving!

```
function configure(cmd, dir)
  dir:match_handler {
    pattern = "^/server-says-hi$",
    file    = "htdocs/config_tests.lua",
    func    = "handle_server_vm",
    scope   = "server"
  }
end
```

State (and Future) of the Wombat

Wombat is a wee baby.
setback: The three folks who had contributed the most **all** had kids.
Stallman would turn in his grave. If he was dead. Having been killed by ninjas.

```
LuaRoot "/www/ac_us_07/"

LuaConfig httpd_config.lua configure
LuaQuickHandler lib/hooks.lua quick
LuaHookTranslateName lib/hooks.lua trans_name
```

Evolving!

```
function configure(cmd, dir)
  dir:match_handler {
    pattern = "^/server-says-hi$",
    file    = "htdocs/config_tests.lua",
    func    = "handle_server_vm",
    scope   = "server"
  }
end
```

Configuration is evolving

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

The way the apache C structs are packaged into Lua will probably change (Thank you Rici Lake!)
How wombat itself is packaged as well!

The URL

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

NO web presence. Seriously. It is evolving quickly so there is just the svn repo. This is it. There are docs, which are kept up to date, in the svn repo.

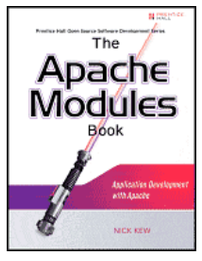
Resources & Docs

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_07.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()
```

Moon Unit is a micro-xUnit test harness which is part of mod_wombat. It is pretty.

```
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 all you need to get this nice test suite!

That's (Really) All Folks!

Brian McCallister

http://kasparov.skife.org/wombat _ac_us_07.pdf