LuaNode

Asynchronous I/O in Lua

Ignacio Burgueño

About Me

Ignacio Burgueño

Software Developer at inConcert

@iburgueno

https://github.com/ignacio

Agenda

- What is Asynchronous I/O?
- What is LuaNode?
- Demo
- Guidelines
- Future Improvements

What does Asynchronous I/O mean?

Asynchronous I/O

It is not only:

- Nonblocking operations

Asynchronous I/O

It is not only:

- Nonblocking operations
- Readyness

Asynchronous I/O

It is not only:

- Nonblocking operations
- Readyness

It is:

- Notification on operation completion

What is LuaNode?

What is LuaNode?

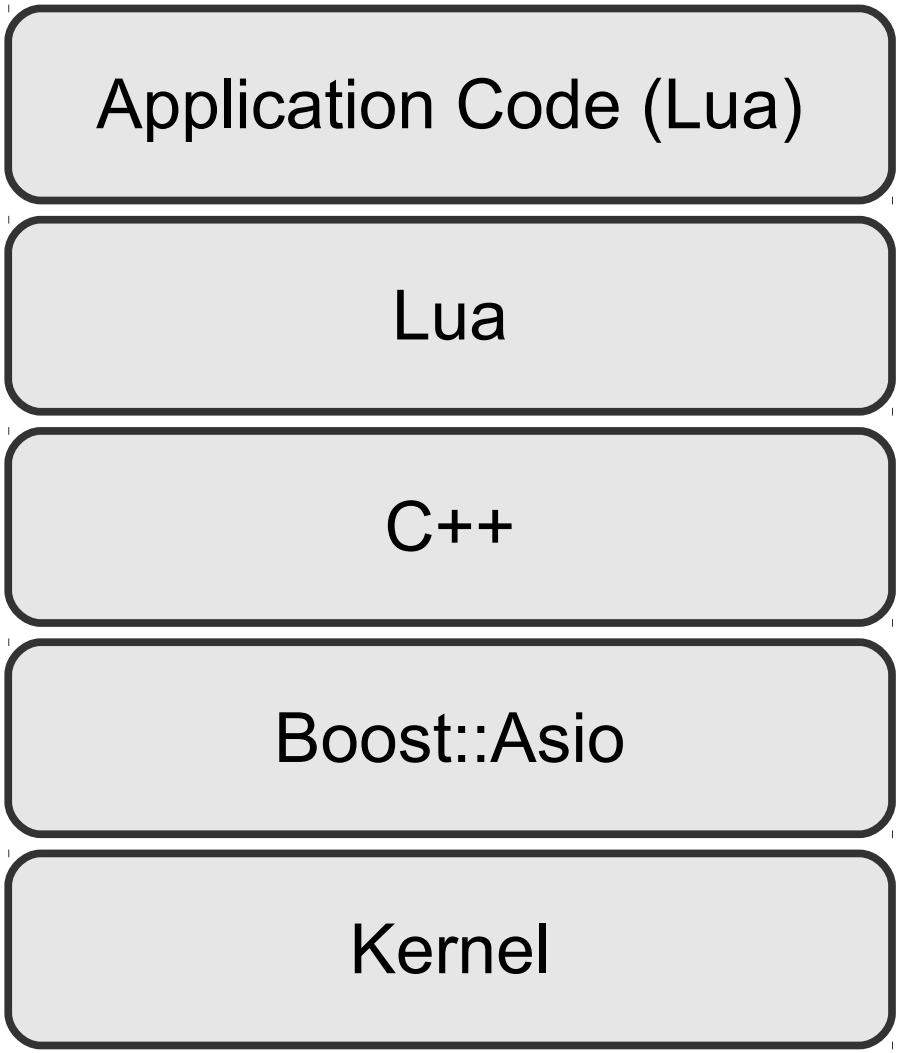
- Event driven, asynchronous IO model. (Like Node.js, nginx)
- Runs Lua code (5.1.5)
- Similar to Node.js, EventMachine (Ruby), Twisted (Python)

https://github.com/ignacio/luanode

What is LuaNode?

JavaScript (v8) + C++ + libuv = Node.js

Lua + C++ + Boost::ASIO = LuaNode


Application Code (Lua)
Lua
C++
Boost::Asio
Kernel

Supported platforms

- Windows (7, 8)
- Linux
- OSX

A taste of LuaNode

```
local function handle_request (server, request, response)
    response:writeHead(200, {["Content-Type"] = "text/plain"})
    response:finish("Hello World!"))
end

luanode.http.createServer(handle_request):listen(8124)
console.log("Server running at http://127.0.0.1:8124/")
process:loop()
```

An example of synchronous IO

```
local socket = require "socket"

local function get (host, file)
    local c = assert(socket.connect(host, 80))
    c:send("GET " .. file .. " HTTP/1.0\r\n\r\n")
    local s = c:receive("*a")
    print("Received ".. #s .. " bytes.")
    c:close()
end

local host = "www.w3.org"

get(host,"/TR/2002/REC-xhtml1-20020801/xhtml1.pdf")

get(host,"/TR/REC-html32.html")
```

Asynchronous I/O example

```
local function get (host, file)
    local c, l = luanode.net.createConnection(80, host), 0
    c:on("connect", function()
        c:on("data", function(self, data) l = l + #data end)
        c:on("close", function()
            console.log("Downloaded file '%s'.  Received %d bytes.", file, l)
        end)
        c:write("GET " .. file .. " HTTP/1.0\r\n\r\n")
    end)
end
get("www.w3.org","/TR/2002/REC-xhtml1-20020801/xhtml1.pdf")
get("www.w3.org","/TR/REC-html32.html")
process:loop()
```

The Loop

- Provided by ASIO
- Handles a queue
- Completed operations are posted there
- Dispatch associated callbacks

Not only IO

Timers

```
setTimeout(function()
    console.log("5 seconds have passed")
end, 5000)
```

Not only IO

Timers

```
setInterval(function()
    console.log("5 seconds have passed")
end, 5000)
```

Not only IO

DNS

```
luanode.dns.lookup("www.google.com",
 function(ok, addrs, hostname)
   console.log(luanode.utils.inspect(addrs))
 end)
```

```
{  family => 4,  port => 0, address => "200.40.0.90"}
```

Not only IO

TTY

```
require "luanode.tty"

local stdin = luanode.tty.ReadStream(Stdio.stdinFD)
stdin:on("data", function(self, data)
    console.log("data:", data)
end)
stdin:resume()
process:loop()
```

Demo time!

Server sent events

- Browser issue two requests that become a persistent one way connection
- The server pushes data to the browser from time to time

Guidelines

- Stay close to Node.js
- Similar API
- Easy to adapt existing libraries

What is available today?

- TCP
- HTTP
- HTTPS
- TIMERS
- TTY
- REPL

When to use?

- Stuff that is IO bound.
- Simple webservers.
- Test servers.
- Mock services.
- Exploratory programming (REPL)
- Load testing.

When not to use?

- Currently, with stuff that is CPU Bound
- (Don't write servers to compute Fibonacci numbers with it)

Some available modules

- Redis (redis-luanode)
- Socket.IO (LuaNode-Socket.IO)
- OAuth (LuaOAuth)
- Route66 (simple url routing, inspired by Orbit)

Existing users (other than me :)

Moonwalk: a Swagger server implementation for Lua.

https://github.com/abadc0de/moonwalk

Xauxi: A reverse proxy

http://ia97lies.github.io/xauxi/

Callback hell

Possible solutions?

- Coroutines
- Futures

Coroutines

```
IssueHttpRequest(function(response)
    -- deal with response
    IssueDBQuery(function(result)
        -- deal with result
        setTimeout(function()
            -- and we are done
        end, 5000)
    end)
end)
```

Coroutines

```
-- makes a fake synchronous function
Sync(function(wait, yield)
    IssueHttpRequest(wait)
    local _, response = yield()
    -- deal with response
    IssueDBQuery(wait)
    local result = yield()
    -- deal with result
    setTimeout(wait, 5000)
    yield()
    -- wait five seconds
    --and we are done
end)    -- end Sync function
```

Coroutines

```
local function Sync(fn)
    local function make_resumer(co)
        return function(...)
            return assert(coroutine.resume(co, ...))
        end
    end
    local co = coroutine.create(fn)
    assert( coroutine.resume(co, make_resumer(co), coroutine.yield ) )
end
```

Future improvements

Future improvements

- Documentation
- Migrate to libuv
- Better multicore support.
- Proper File I/O
- UDP support
- Better coroutine integration
- Multithreading

LuaNode makes writing async code easy.

Questions?

Thanks!

Ignacio Burgueño - @iburgueno