NGINX stream 子系统的简介以及 OpenResty 对其的支持

孙大同 (@dndx)

OpenResty Con 2017 北京

2017 年 10 月 21 日

关于我

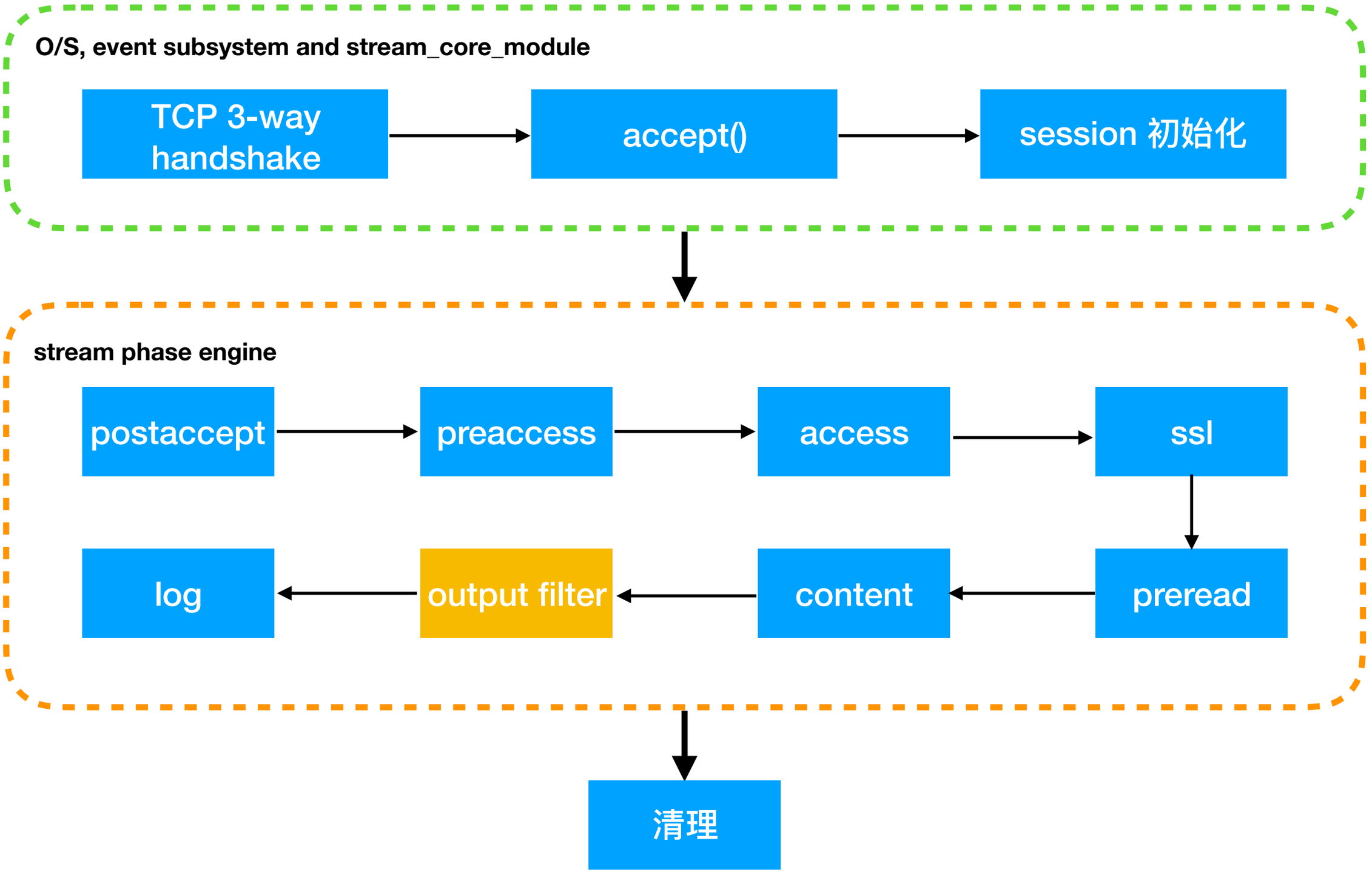
- 90 后
- 曾在 LinkedIn，Cloudflare 等公司工作
- OpenResty Inc. 技术合伙人
- 热爱钻研底层技术以及折腾
- 飞行爱好者。拥有固定翼私人飞行执照，仪表飞行资质以及无人机驾驶执照。

NGINX stream 子系统的简介

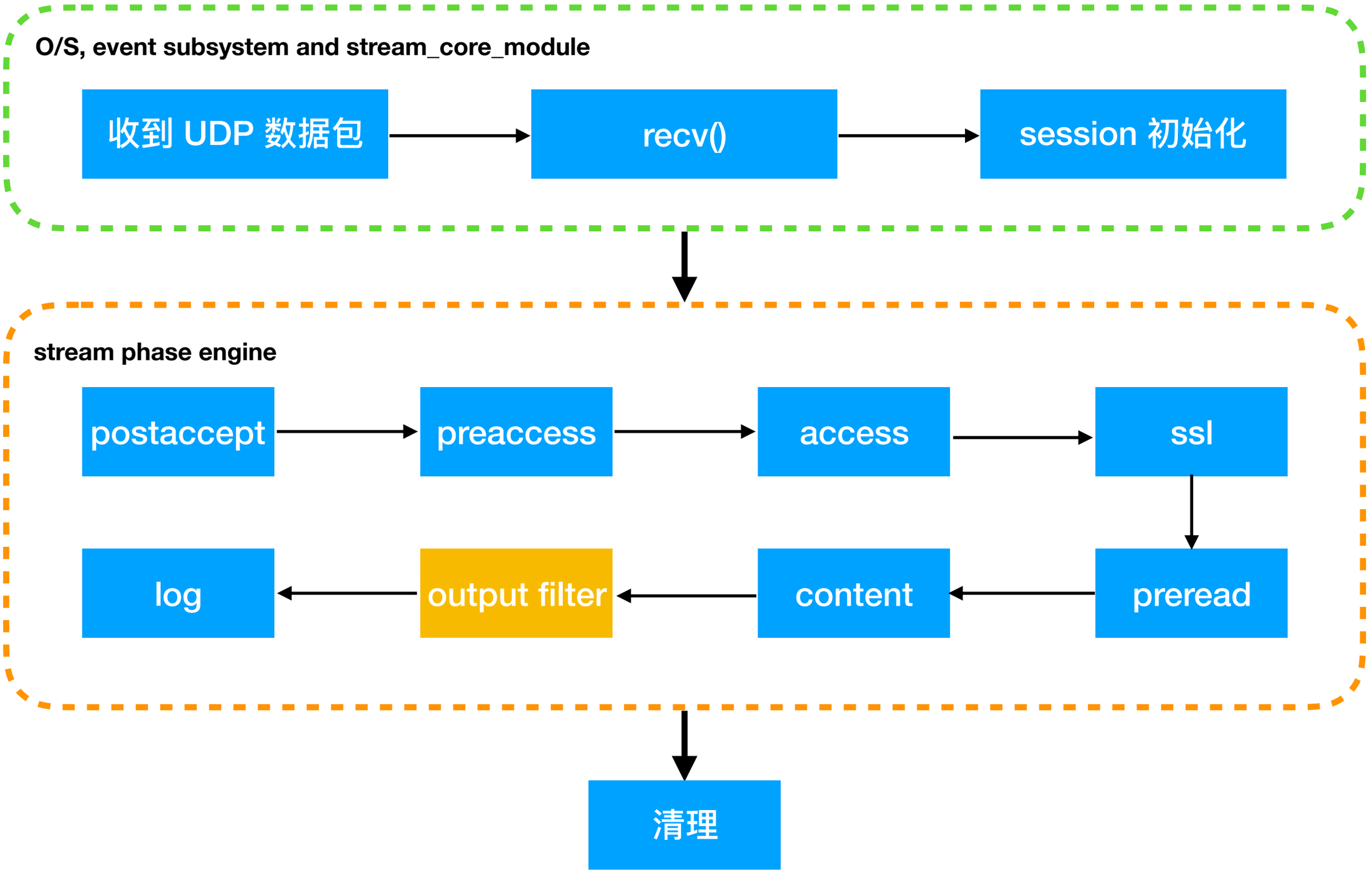
NGINX stream 子系统的历史

- 从 2015 年 4 月发布的 v1.9.0 开始正式出现
- 用于实现通用的 TCP 以及 UDP 服务器
- 功能在 v1.11.2 - v1.11.8 之间得到了很大的提升
- v1.11.x 版本的 stream，核心变化十分巨大且不向后兼容
- 截止到 v1.13.x，核心部分已经趋于稳定
- OpenResty 早在 2016 年 1 月就有了实验性的支持

NGINX stream 子系统的连接处理 (TCP)



NGINX stream 子系统的连接处理 (UDP)



stream-lua-nginx-module 对最新 NGINX 核心的支持

- Kong Inc. 赞助了 stream-lua-nginx-module 的开发
- 完全基于最新的 lua-nginx-module 代码重写
- 重写过程中加入了对新的 phase 系统和变量系统的支持
- 增加了新的 API 来适应 stream 模块的特殊需求
- 新的 meta-lua-nginx-module

no “location” block

preread_by_lua

- 访问控制
- 高级代理策略
- 可以使用会 yield 的 API
- 可以向客户输出内容
- 不受 preread_buffer_size 和 preread_timeout 的限制

balancer_by_lua

- API 于 lua-nginx-module 现有的完全兼容
- 需要在 stream 的 block 下单独定义 upstream
- state_name, **status_code** = balancer.get_last_failure()
- balancer.set_timeouts(connect_timeout, **send_timeout**, **read_timeout**)

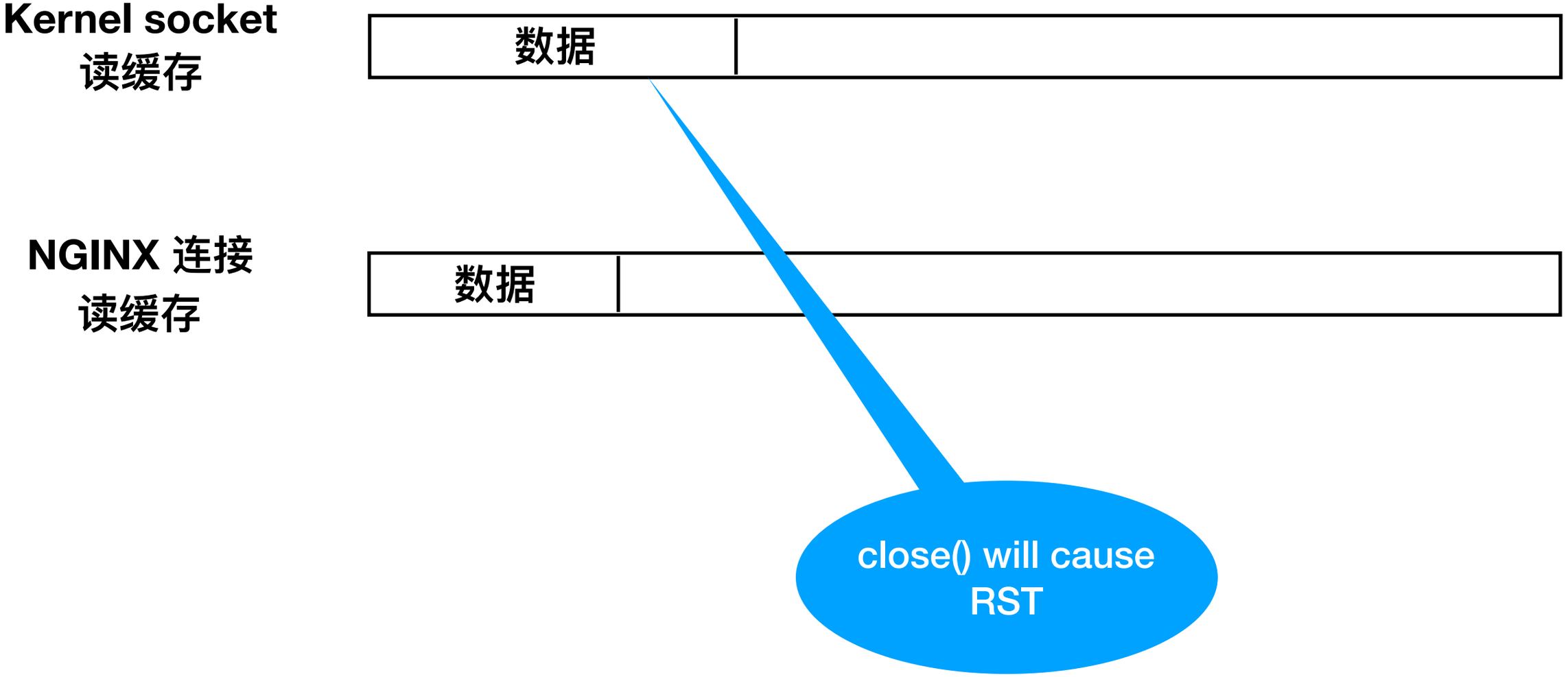
log_by_lua

- 发生在连接即将被关闭时
- 不能 yield
- 可以使用 ngx.timer.* API
- 可以访问所有的变量以及 ngx.ctx

stream-lua-nginx-module 使用过程中的注意事项

- 避免频繁在请求的内存池上分配
- UDP 子系统对数据包的特殊处理
- 不要在没有完全清空操作系统的 socket 读缓冲区的情况下关闭连接
  - 新的 API：tcpsock:shutdown() 用来模拟 lingering close

Lingering close



tcpsock:shutdown

```
local LINGERING_TIME = 30 -- 30 seconds
local LINGERING_TIMEOUT = 5000 -- 5 seconds

local ok, err = sock:shutdown("send")
if not ok then
    ngx.log(ngx.ERR, "failed to shutdown: ", err)
    return
end

local deadline = ngx.time() + LINGERING_TIME

sock:settimeouts(nil, nil, LINGERING_TIMEOUT)

repeat
    local data, _, partial = sock:receive(1024)
until (not data and not partial) or ngx.time() >= deadline
```

Source: https://github.com/openresty/stream-lua-nginx-module#tcpsockshutdown

stream-lua-nginx-module 的开发思路

- I/O 主要依靠于 ngx.req.socket，只支持 raw socket
- 有少量的 ngx.say()，ngx.print() 的支持
- 可以利用 preread_by_lua，proxy 模块和 balancer_by_lua 来高效的转发

stream-lua-nginx-module 未来的功能

- ssl_certificate_by_lua*
- access_by_lua*
- shdict 跨子系统的共享
- lua-resty-core 的集成

例子：echo 服务器

```
stream {
    server {
        listen 1234;

        content_by_lua_block {
            local sock, err = ngx.req.socket()
            if not sock then
                ngx.exit(500)
            end

            while true do
                local data
                data, err = sock:receive('*l')
                if not data then
                    return ngx.exit(200)
                end

                local sent
                sent, err = sock:send(data .. '\n')
                if err then
                    return ngx.exit(200)
                end
            end
        }
    }
}
```

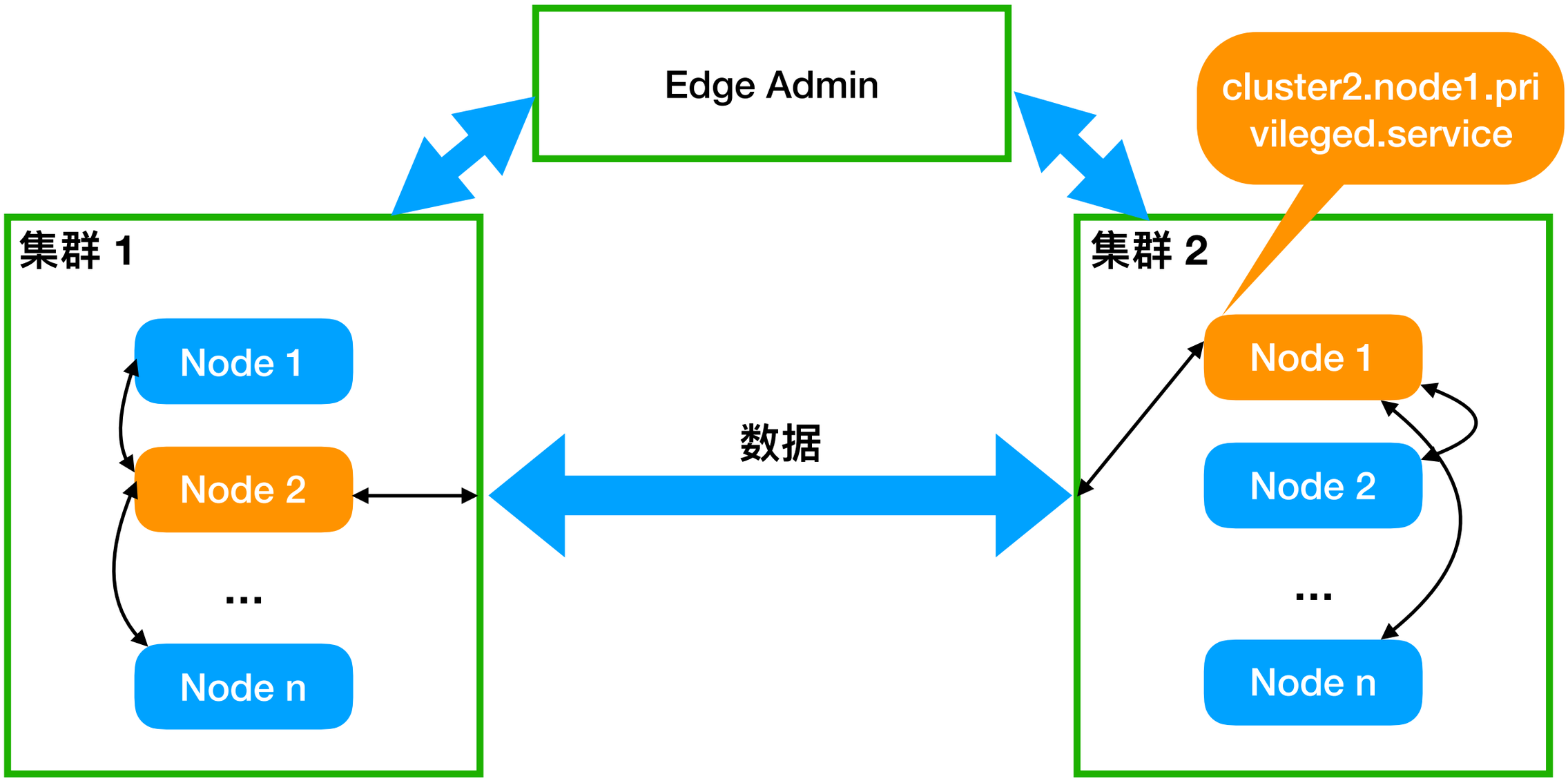
Demo: telnet 聊天室

https://github.com/dndx/telnet-chat

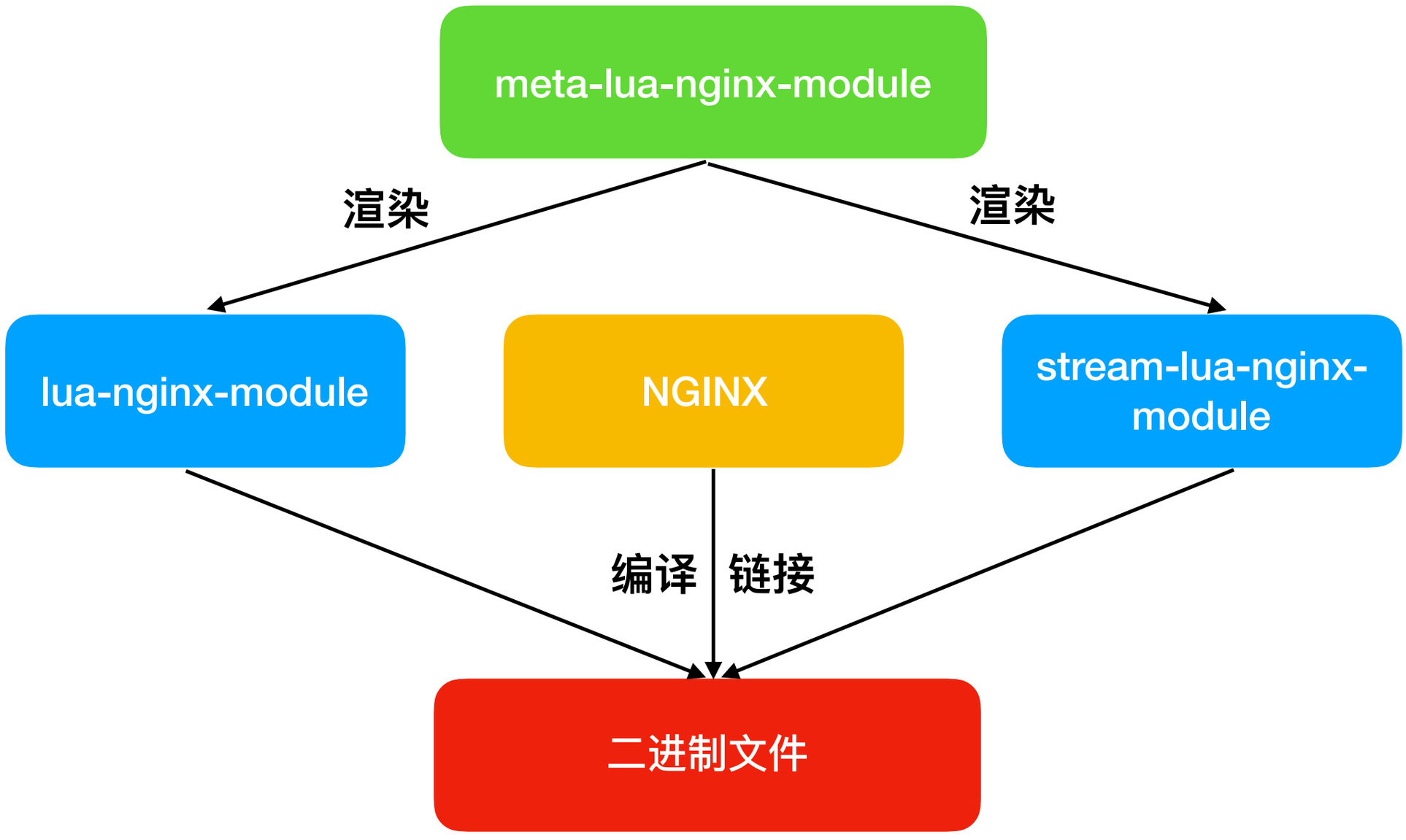
stream-lua-nginx-module 模块在 OpenResty Edge 2 中的应用

- 已经在用
  - 类 memcached 协议的服务
  - DNS 服务器
- 很快会有
  - 基于规则的四层流量代理
  - Routing Platform

OpenResty Edge 2 Routing Platform



代码生成过程


meta-lua-nginx-module
渲染
渲染
lua-nginx-module
NGINX
stream-lua-nginx-module
编译
链接
二进制文件

更多信息

- https://github.com/openresty/stream-lua-nginx-module
- https://github.com/openresty/meta-lua-nginx-module
- https://openresty.com
- Email: datong#openresty.com
- 欢迎贡献 PR!

Questions?

Thank You!

A machine coding company