一起动手学OR新特性

@Gerrard

内容

ngx.semaphore 信号量同步

ngx.shared.DICT 共享内存list

balancer_by_lua* 用lua写负载均衡

stream-lua-nginx-module lua高性能tcp服务

resty 相关工具

实验环境 : nginx version: openresty/1.11.2.1
stream-lua-nginx-module: commit 75e0b16
相关代码 : https://github.com/openresty-suzhou/meetup

信号量ngx.semaphore

- 高效
- 进程内 跨context

文档 https://github.com/openresty/lua-resty-core/blob/master/lib/ngx/semaphore.md

生产者消费者问题

场景

- Coroutine A 生产数据 送入buffer
- Coroutine B 消费数据 从buffer取数据

原来的方式

Coroutine B 不断地polling + nginx.sleep

缺点

仍可能有毫秒级别的延迟

使用ngx.semaphore机制

```
local co = ngx.thread.spawn(produce, buffer)
local ok, err = sema:wait(1) --wait at most 1s
if ok then
    if buffer.message then
        ngx.say('main thread : receive message : ', buffer.message)
    else
        ngx.say('main thread : unexpected error, got semaphore but no message in buffer')
    end
else
    ngx.say('main thread : failed to wait on sema: ', err)
end
ngx.say('main thread : end.')
```

```
local function produce(buffer)
    ngx.say('producer thread : producting data ...')
    ngx.sleep(0.1) --wait a little time
    buffer.message = 'This is a message from producer!' --producing data
    ngx.say('producer thread : posting to sema...')
    sema:post(1)
end
```

```
gerrard@ubuntu:~/workspace/test$  curl 127.0.0.1:8000/semaphore
main thread : no message in buffer, waiting for semaphore from producer...
producer thread : producting data ...
producer thread : posting to sema...
main thread : receive message : This is a message from producer!
main thread : end.
```

ngx.shared.DICT 新增API

- lpush
- lpop
- rpush
- rpop
- llen

文档 https://github.com/openresty/lua-nginx-module#ngxshareddict

像redis的list？

支持类型

- 字符串
- 数字

使用场景

- 轻松实现跨worker的stack, queue, deque

借助lpush,lpop封装一个简单的栈

```
function _M.new(self, name)
    self.name = name
    self.list = ngx.shared.list
    return setmetatable({},{__index =
_M})
end

function _M.push(self,v)
    return self.list:lpush(self.name, v)
end

function _M.pop(self)
    return self.list:lpop(self.name)
end

function _M.size(self)
    return self.list:llen(self.name)
end
```

借助lpush,lpop封装一个简单的栈

```
local myStack = stack:new('stack')
for k, v in pairs(myStack) do
    print(k, tostring(v))
end

local elements = {'first', 2,true,4.001}

for _, v in ipairs(elements) do
    local len, err = myStack:push(v)
    if len then
        ngx.say('pushed value : ' .. v .. ' to stack,
stack length : ' .. len )
    else
        ngx.say('pushed to stack falied, err : ', err)
    end
end
```

```
while true do
    local val, err = myStack:pop()
    if not val then
        ngx.say('pop from stack falied, err : ', err)
        break
    end
    ngx.say('pop from stack, value : ' .. val .. ', sizeof
stack : ', myStack:size())
end
```

```
gerrard@ubuntu:~/workspace/test$  curl 127.0.0.1:8000/shdict
pushed value : first to stack, stack length : 1
pushed value : 2 to stack, stack length : 2
pushed to stack falied, err : bad value type
pushed value : 4.001 to stack, stack length : 3
pop from stack, value : 4.001, sizeof stack : 2
pop from stack, value : 2, sizeof stack : 1
pop from stack, value : first, sizeof stack : 0
pop from stack falied, err : nil
```

负载均衡 balancer_by_lua*

- 动态选择upstream 细化到每个请求
- 灵活，自定义策略

文档 https://github.com/openresty/lua-resty-core/blob/master/lib/ngx/balancer.md

upstream

原生upstream

- 手动配置，无法动态调整
- 策略有限 RR, weight，IP hash

使用场景

- 动态调整upstream
- 自定义负载均衡策略

实现一个简单的RR

```
local balancer = require 'ngx.balancer'
local host = '127.0.0.1'
local port = {9000, 9001, 9002}
local val, err = ngx.shared.list:incr('count', 1, -1)
if not val then
    ngx.log(ngx.ERR, 'failed to incr key "count", err : ', err)
    ngx.exit(500)
end

local state, status = balancer.get_last_failure()
if state then
    ngx.log(ngx.ERR,'last peer failure:', state, " ", status)
end
```

```
if not ngx.ctx.tries then
    ngx.ctx.tries = 0
end
if ngx.ctx.tries < 1 then
    local ok, err = balancer.set_more_tries(1)
    if not ok then
    return error('failed to set more tries: ', err)
    elseif err then
    ngx.log(ngx.ERR, "set more tries:", err)
    end
end
ngx.ctx.tries = ngx.ctx.tries + 1
local ok, err = balancer.set_current_peer(host,
port[val%3+1])
if not ok then
    ngx.log(ngx.ERR, 'failed to set the current peer : ', err)
    return ngx.exit(500)
end
```

实现一个简单的RR

```
server {
    listen 8080;
    location / {
        proxy_pass http://backend/;
        proxy_next_upstream_tries 2;
    }
}
upstream backend{
        server 1.2.3.4;
        balancer_by_lua_file src/balancer.lua;
        keepalive 60;
    }
```

```
server {
        listen 127.0.0.1:9000;
        location / {
            echo "upstream from port 9000...";
        }
    }
    server {
        listen 127.0.0.1:9001;
        location / {
            echo "upstream from port 9001...";
        }
    }
    server {
        listen 127.0.0.1:9002;
        location / {
            echo "upstream from port 9002...";
        }
    }
```

nginx stream

- 2015-04-28 nginx-1.9.0 mainline version has been released, with the stream module for **generic TCP/UDP proxying and load balancing**.

http://nginx.org/en/docs/stream/ngx_stream_core_module.html

- 在这个特性之前nginx只能做7层的应用层代理, 4层的TCP代理一般用LVS, Haproxy等完成

nginx stream

```
# stream的配置示例, 分别是udp proxy, tcp proxy, unix domain proxy
stream {
    upstream backend {
        hash $remote_addr consistent;
        server backend1.example.com:12345 weight=5;
        server 127.0.0.1:12345            max_fails=3 fail_timeout=30s;
        server unix:/tmp/backend3;
    }
    upstream dns {
       server 192.168.0.1:53535;
       server dns.example.com:53;
    }
    server {
        listen 12345;
        proxy_connect_timeout 1s;
        proxy_timeout 3s;
        proxy_pass backend;
    }
    server {
        listen 127.0.0.1:53 udp;
        proxy_responses 1;
        proxy_timeout 20s;
        proxy_pass dns;
    }
    server {
        listen [::1]:12345;
        proxy_pass unix:/tmp/stream.socket;
    }
}
```

stream块ngx_stream_core_module

- nginx stream块的lua扩展
- 目前还处于开发阶段, 还没有正式发布, 需要自已下载安装和体验

文档 https://github.com/openresty/stream-lua-nginx-module

ngx_stream_core_module

预计完成如下特性后会正式发布

- support access_by_lua_block
- access_by_lua_file
- support log_by_lua_block
- log_by_lua_file
- support balancer_by_lua_block
- balancer_by_lua_file
- support ssl_certificate_by_lua_block
- ssl_certificate_by_lua_file
- support ngx.semaphore
- add ngx_meta_lua_module
- support lua-resty-core
- add lua_postpone_output

ngx_stream_core_module性能测试

Stream块配置

```
stream {
    server {
        listen 1235;
        content_by_lua_block {
                local sock = assert(ngx.req.socket(true))
                local data = sock:receive()
                ngx.say('hello world')
        }
    }
}
```

```
ab -n 10000 -c  100 -k http://127.0.0.1:1235/
```

ngx_stream_core_module性能测试

```
Server Hostname:        127.0.0.1
Server Port:            1235
Document Path:          /
Document Length:        0 bytes
Concurrency Level:      100
Time taken for tests:   0.406 seconds
Complete requests:      10000
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      120000 bytes
HTML transferred:       0 bytes
Requests per second:    24635.64 [#/sec] (mean)
Time per request:       4.059 [ms] (mean)
Time per request:       0.041 [ms] (mean, across all concurrent
requests)
Transfer rate:          288.70 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2   0.7      2       7
Processing:     1    2   0.8      2       9
Waiting:        0    2   0.7      2       8
Total:          2    4   1.1      4      12
```

stream lua 的echo 服务器

```
server {
        listen 1234;
        content_by_lua_block {
            local sock = ngx.req.socket()
            while (true) do
                local line, err = sock:receive('*l')  -- read a line from downstream
                            if not line then
                                    ngx.say(err)
                                    break
                            elseif line == 'exit' then
                    ngx.say('bye')
                    break
                end
                ngx.say(line) -- output data
            end
        }
    }
```

基于luasocket的客户端

```
local socket = require'socket'
local host = '127.0.0.1'
local port = 1234
local sock = assert(socket.connect(host, port))
sock:settimeout(0)

print('please input anything, exit to quit:')
local input, recvt, sendt, status
local connected = true
```

```
gerrard@ubuntu:~/workspace/test/client$ lua test.lua
please input anything, exit to quit:
hello world
hello world
1234567
1234567
exit
bye
```

```
while connected do
    input = io.read()
    if #input > 0 then
        assert(sock:send(input .. '\n'))
    end
    recvt, sendt, status = socket.select({sock}, nil, 0.1)
    while #recvt > 0 do
        local response, reveive_status = sock:receive()
        if reveive_status ~= 'closed' then
            if response then
                print(response)
                if response == 'bye' then
                    sock:close()
                    connected = false
                end
            end
        end
        break
    end
end
```

文档查看工具restydoc

- 可以按模块名搜索，也可以按节名搜，或者二者的组合。
- 搜索方式是精确匹配优先，前缀次之，包含最次

```
gerrard@ubuntu:~/workspace/test/src$ restydoc  -h
Usage:
    /home/gerrard/workspace/test/openresty/bin/restydoc [options] [module]

Options:
    -h              Print this help.
    -s SECTION      Specify the section name to be searched

For bug reporting instructions, please see:

    <https://openresty.org/en/community.html>

Copyright (C) Yichun Zhang (agentzh). All rights reserved.
```

use case

restydoc -s content_by_lua_file

```
ngx_lua-0.10.6(7)              ngx_lua-0.10.6              ngx_lua-0.10.6(7)


   content_by_lua_file
       syntax: content_by_lua_file <path-to-lua-script-file>

       context: location, location if

       phase: content

       Equivalent to content_by_lua, except that the file specified by
       "<path-to-lua-script-file>" contains the Lua code, or, as from the
       "v0.5.0rc32" release, the "Lua/LuaJIT bytecode" to be executed.

       Nginx variables can be used in the "<path-to-lua-script-file>" string
       to provide flexibility. This however carries some risks and is not
       ordinarily recommended.

       When a relative path like "foo/bar.lua" is given, they will be turned
       into the absolute path relative to the "server prefix" path determined
       by the "-p PATH" command-line option while starting the Nginx server.

       When the Lua code cache is turned on (by default), the user code is
       loaded once at the first request and cached and the Nginx config must
       be reloaded each time the Lua source file is modified.  The Lua code
       cache can be temporarily disabled during development by switching
       lua_code_cache "off" in "nginx.conf" to avoid reloading Nginx.

       Nginx variables are supported in the file path for dynamic dispatch,
       for example:

            # WARNING: contents in nginx var must be carefully filtered,
            # otherwise there'll be great security risk!
            location ~ ^/app/([-_a-zA-Z0-9/]+) {
                set $path $1;
                content_by_lua_file /path/to/lua/app/root/$path.lua;
            }

       But be very careful about malicious user inputs and always carefully
```

restydoc resty.redis

```
lua-resty-redis-0.25(7)      lua-resty-redis-0.25      lua-resty-redis-0.25(7)



Name
       lua-resty-redis - Lua redis client driver for the ngx_lua based on the
       cosocket API

       This library is considered production ready.

Description
       This Lua library is a Redis client driver for the ngx_lua nginx module:

       https://github.com/openresty/lua-nginx-module/#readme

       This Lua library takes advantage of ngx_lua's cosocket API, which
       ensures 100% nonblocking behavior.

       Note that at least ngx_lua 0.5.14 <https://github.com/chaoslawful/lua-
       nginx-module/tags> or OpenResty 1.2.1.14
       <http://openresty.org/#Download> is required.

Synopsis
               # you do not need the following line if you are using
               # the OpenResty bundle:
               lua_package_path "/path/to/lua-resty-redis/lib/?.lua;;";

               server {
                   location /test {
                       content_by_lua '
                           local redis = require "resty.redis"
                           local red = redis:new()

                           red:set_timeout(1000) -- 1 sec

                           -- or connect to a unix domain socket file listened
                           -- by a redis server:
                           --     local ok, err = red:connect("unix:/path/to/redis.so

                           local ok, err = red:connect("127.0.0.1", 6379)
:
```

命令行工具resty

- 像lua和luajit一样的方式运行openresty的lua脚本
- 通过init_worker_by_lua初始化lua代码然后在ngx.timer回调中运行

文档 https://github.com/openresty/resty-cli

usage

```
gerrard@ubuntu:~/workspace/test$ resty -h
resty [options] [lua-file [args]]

Options:
    -c num              Set maximal connection count (default: 64).
    -e prog             Run the inlined Lua code in "prog".
    --help              Print this help.

    --http-include path Include the specified file in the nginx http configuration bloc
                        (multiple instances are supported).

    -I dir              Add dir to the search paths for Lua libraries.

    --main-include path Include the specified file in the nginx main configuration bloc
                        (multiple instances are supported).

    --nginx             Specify the nginx path (this option might be removed in the futu
    -V                  Print version numbers and nginx configurations.
    --valgrind          Use valgrind to run nginx
    --valgrind-opts     Pass extra options to valgrind

For bug reporting instructions, please see:

    <https://openresty.org/en/community.html>

Copyright (C) Yichun Zhang (agentzh). All rights reserved.
```

参考

- 官网 https://openresty.org/
- Github https://github.com/openresty
- 邮件列表(推荐，需翻墙)

https://groups.google.com/forum/#!forum/openresty

The End

Thanks!