APISIX 高性能实践

--by Yuansheng

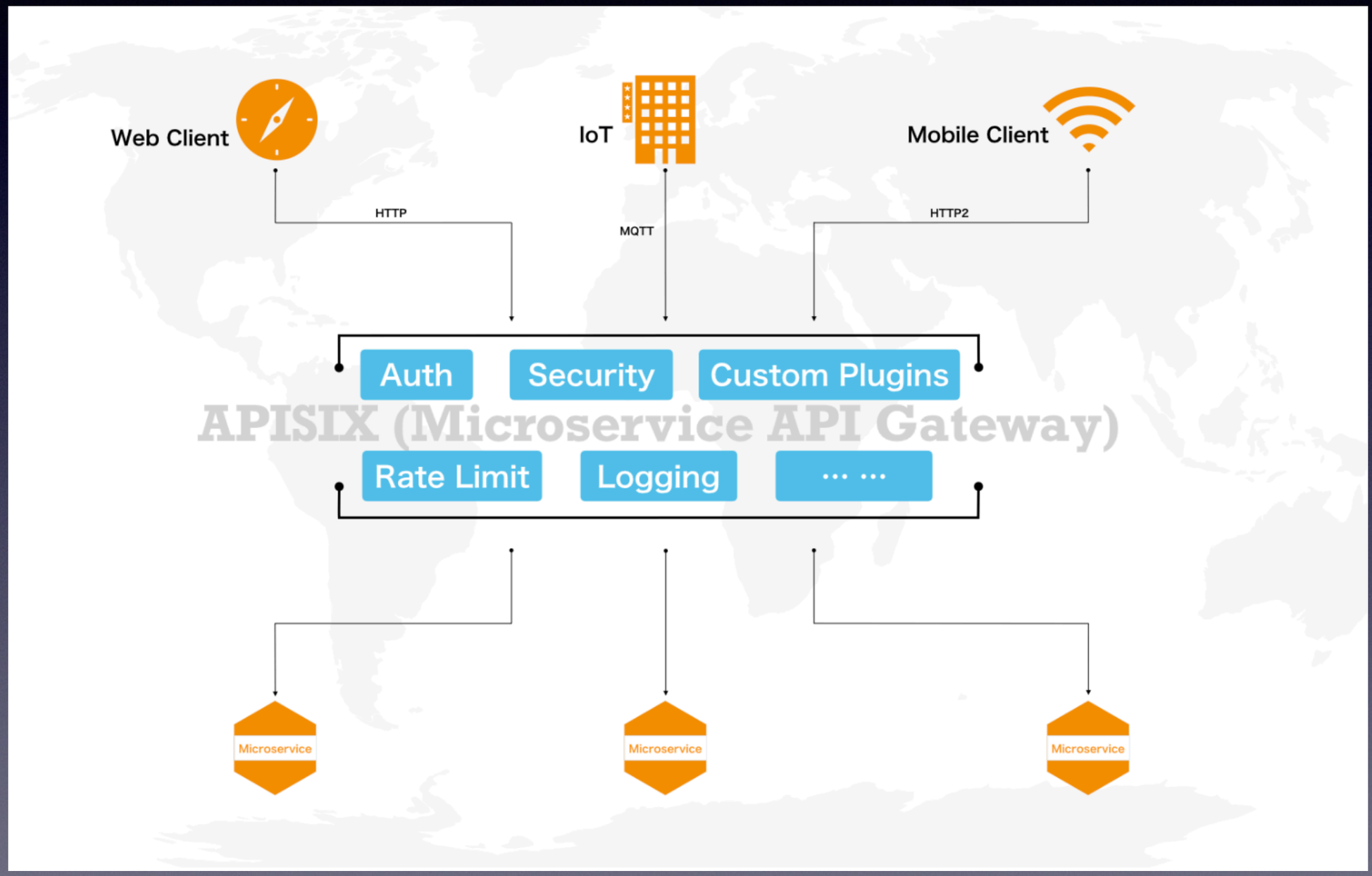
王院生

2014 入职 360 初识 OpenResty，空闲时间写了《OpenResty 最佳实践》聚集国内开发者
2017 作为技术合伙人加入 OpenResty Inc.
2019 年工作重心放到开源工作，并开始 APISIX 征程

我理想公司的宗旨

“依托开源社区，致力于微服务 API 相关技术的创新和实现”

什么是 API 网关


Web Client
IoT
Mobile Client
HTTP
MQTT
HTTP2
Auth
Security
Custom Plugins
APISIX (Microservice API Gateway)
Rate Limit
Logging
… …
Microservice
Microservice
Microservice

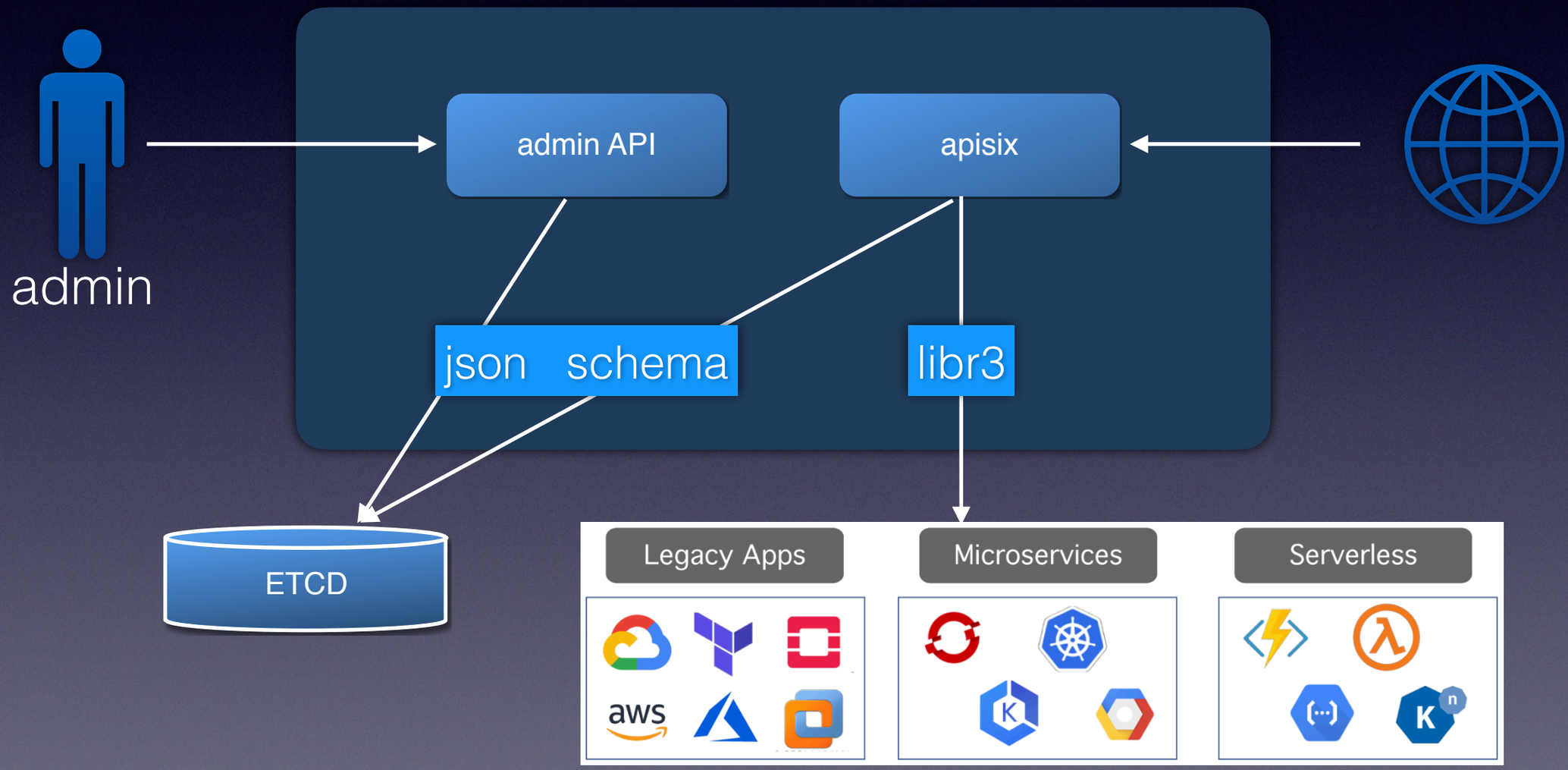
微服务 API 网关

- 动态更新
- 更低延迟
- 用户自定义插件
- 更集中的管理 API

微服务 API 网关

- 通过社区聚焦
- 简洁的 core
- 可扩展
- 顶级性能
- 低延迟

APISIX 架构


admin API
apisix
admin
json schema
libr3
ETCD
Legacy Apps
Microservices
Serverless
aws

APISIX 已有功能

- Cloud-Native
- Dynamic Load Balancing
- Hash-based Load Balancing
- SSL
- Monitoring
- Forward Proxy
- Authentications
- Limit-rate
- Limit-count
- Limit-concurrency
- CLI
- REST API
- Clustering
- Scalability
- High performance
- Custom plugins

入驻 CNCF 家族

https://landscape.cncf.io/category=api-gateway&format=card-mode&grouping=category

CLOUD NATIVE Landscape

CNCF Cloud Native Interactive Landscape

CLOUD NATIVE COMPUTING FOUNDATION

Reset Filters

The Cloud Native Trail Map (png, pdf) is CNCF's recommended path through the cloud native landscape. The cloud native landscape (png, pdf) and serverless landscape (png, pdf) are dynamically generated below. Please open a pull request to correct any issues. Greyed logos are not open source. Last Updated: 2019-07-05 13:52:08Z

Grouping: Category
Sort By: Alphabetical (a to z)
Category: API Gateway
CNCF Relation: Any
License: Any
Organization: Any
Headquarters Location: Any

You are viewing 12 cards with a total of 42,886 stars, market cap of $726.47B and funding of $84.35M.

Landscape | Card Mode | Serverless

Tweet 682

Example filters:

- Cards by age
- Open source landscape
- Cards in categories
- Cards by stars
- Cards from China
- Certified K8s/KCSP/KTP
- Cards by MCap/Funding

Download as CSV

Orchestration & Management - API Gateway (12)

3Scale ★159 Red Hat MCap: $33.43B	**Ambassador** ★2,026 Datawire Funding: $250K	**APISIX** ★335 APISIX	**Express Gateway** ★1,643 LunchBadger Funding: $1.5M
Gloo ★1,507 Solo.io Funding: $13.5M	**Kong** ★22,337 Kong Funding: $69.1M	**KrakenD** ★1,642 Brutale	**Mia-Platform** Mia-Platform

APISIX 状态

- 目前最新版本 0.5，架构：ETCD + libr3 + rapidjson。
- 超 70% 的代码覆盖率。
- 核心代码覆盖率超过 90%。
- 有可能是性能最高的 API 网关。
- 下版本将自带管理界面，降低入门门槛。

APISIX 的性能

VS

```lua
function _M.http_access_phase()
    local uri = ngx.var.uri
    local host = ngx.var.host
    local method = ngx.req.get_method()
    local remote_addr = ngx.var.remote_addr
    fake_fetch(uri, host, method, remote_addr)
end

function _M.http_header_filter_phase()
    if ngx.ctx then
        -- do something
    end
end

function _M.http_log_phase()
    if ngx.ctx then
        -- do something
    end
end

function _M.http_admin()
end

function _M.http_ssl_phase()
    if ngx.ctx then
        -- do something
    end
end
```

性能只下降 15%

单 worker：23-24k 的 QPS

4 worker：68k 的 QPS

平台：alicloud ecs.ic5.3xlarge

奇技淫巧

OpenResty 编程哲学

- 大事化小，小事化了
- 尽可能少的少创建临时对象
- 对速度有要求，还是 C/C++

奇技淫巧

- delay_json
- 前缀树 vs HASH vs 遍历
- uri dispatch vs host dispatch
- ngx.log 是个 NYI
- gc for cdata
- gc for lua table

奇技淫巧

- 如何保护好生命周期很长的 cdata 对象
- ngx.var.* 是比较慢的
- 减少每请求的垃圾对象
- lrucache 的使用技巧

delay_json

```lua
local delay_tab = setmetatable({data = "",
                                force = false}, {
    __tostring = function(self)
        return encode(self.data, self.force)
    end
})

function _M.delay_encode(data, force)
    delay_tab.data = data
    delay_tab.force = force
    return delay_tab
end
```

```
core.log.info(core.json.delay_encode(api_ctx.matched_route))
```

前缀树 vs HASH vs 遍历

- 前缀树: 借助 libr3 完成前缀等高级匹配，比如：uri，host
- HASH：完成静态字符串匹配
- 遍历：永远都是糟糕的

ngx.log 是 NYI

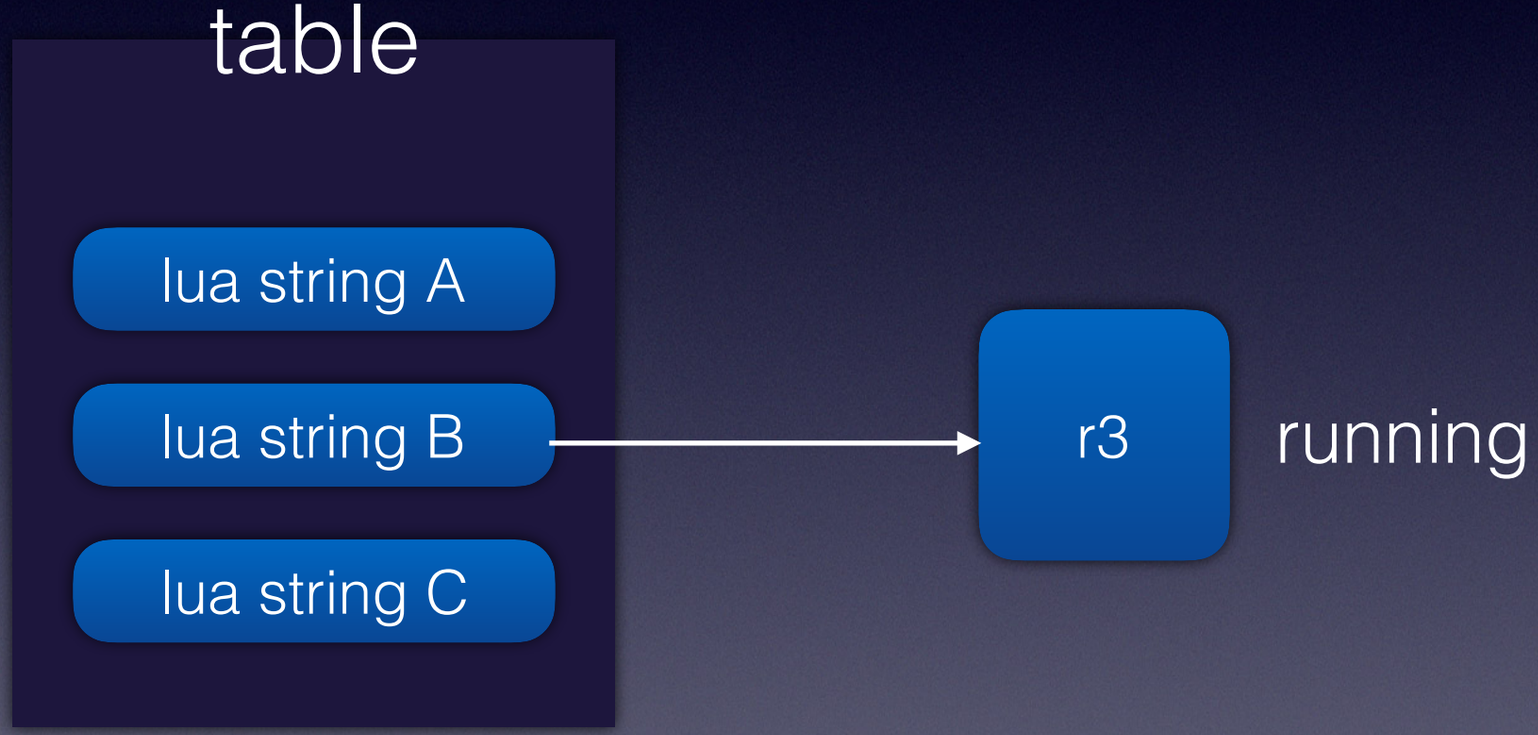
```
for name, log_level in pairs({stderr = ngx.STDERR,
                              emerg  = ngx.EMERG,
                              alert  = ngx.ALERT,
                              crit   = ngx.CRIT,
                              error  = ngx.ERR,
                              warn   = ngx.WARN,
                              notice = ngx.NOTICE,
                              info   = ngx.INFO, }) do
    _M[name] = function(...)
        if cur_level and log_level > cur_level then
            return
        end

        return ngx_log(log_level, ...)
    end
end
```

gc for cdata and table

```lua
-- only work under lua51 or luajit
local function setmt__gc(t, mt)
    local prox = newproxy(true)
    getmetatable(prox).__gc = function() mt.__gc(t) end
    t[prox] = true
    return setmetatable(t, mt)
end
```

场景：worker 进程退出或放入 lrucache 的对象，可能是 cdata 对象，比如要调用 free 才能真正释放。

如何保护常驻内存的 cdata 对象



不会增加计数引用

ngx.var.* 是比较慢的

原因：C 不支持动态，必须通过一次 hash 查找
解决办法：github.com/iresty/lua-var-nginx-module

```
ngx_int_t
ngx_http_lua_var_ffi_remote_addr(ngx_http_request_t *r, r
{
    remote_addr->len = r->connection->addr_text.len;
    remote_addr->data = r->connection->addr_text.data;

    return NGX_OK;
}
```

减少每请求的垃圾对象

- 减少不必要的字符串拼接
- table 缓存

减少每请求的垃圾对象

```
function _M.http_log_phase()
    local api_ctx = run_plugin("log")
    if api_ctx then
        if api_ctx.uri_parse_param then
            core.tablepool.release("uri_parse_param", api_
        end

        core.ctx.release_vars(api_ctx)
        if api_ctx.plugins then
            core.tablepool.release("plugins", api_ctx.plug
        end

        core.tablepool.release("api_ctx", api_ctx)
    end
end
```

lrucache 的使用技巧

- lrucache 的二次封装
- key 要尽量短、简单
- version 可降低垃圾缓存
- 重用 stale 状态的缓存数据

lrucache 的使用技巧

```lua
    local empty_tab = {}
    local route_items
local function create_r3_router(routes)
    ...

    local r3 = r3router.new(route_items)
    r3:compile()
    return r3
end


function _M.get()
    return core.lrucache.global("/routes", routes.conf_version,
                                create_r3_router, routes.values)
end
```

lrucache 的使用技巧

```lua
return function (key, version, create_obj_fun, ...)
    local obj, stale_obj = lru_obj:get(key)
    if obj and obj._cache_ver == version then
        return obj.val
    end

    if stale_obj and stale_obj._cache_ver == version then
        lru_obj:set(key, obj, item_ttl)
        return stale_obj
    end

    local err
    obj, err = create_obj_fun(...)
    obj._cache_ver = version
    lru_obj:set(key, obj, item_ttl)

    return obj, err
end
```

其他

- github.com/iresty/apisix
- 更多技巧：lua/apisix/core
- QQ 交流群：552030619
- 欢迎大家参与并使用
- 提供一对一的服务