

APISIX 高性能实践

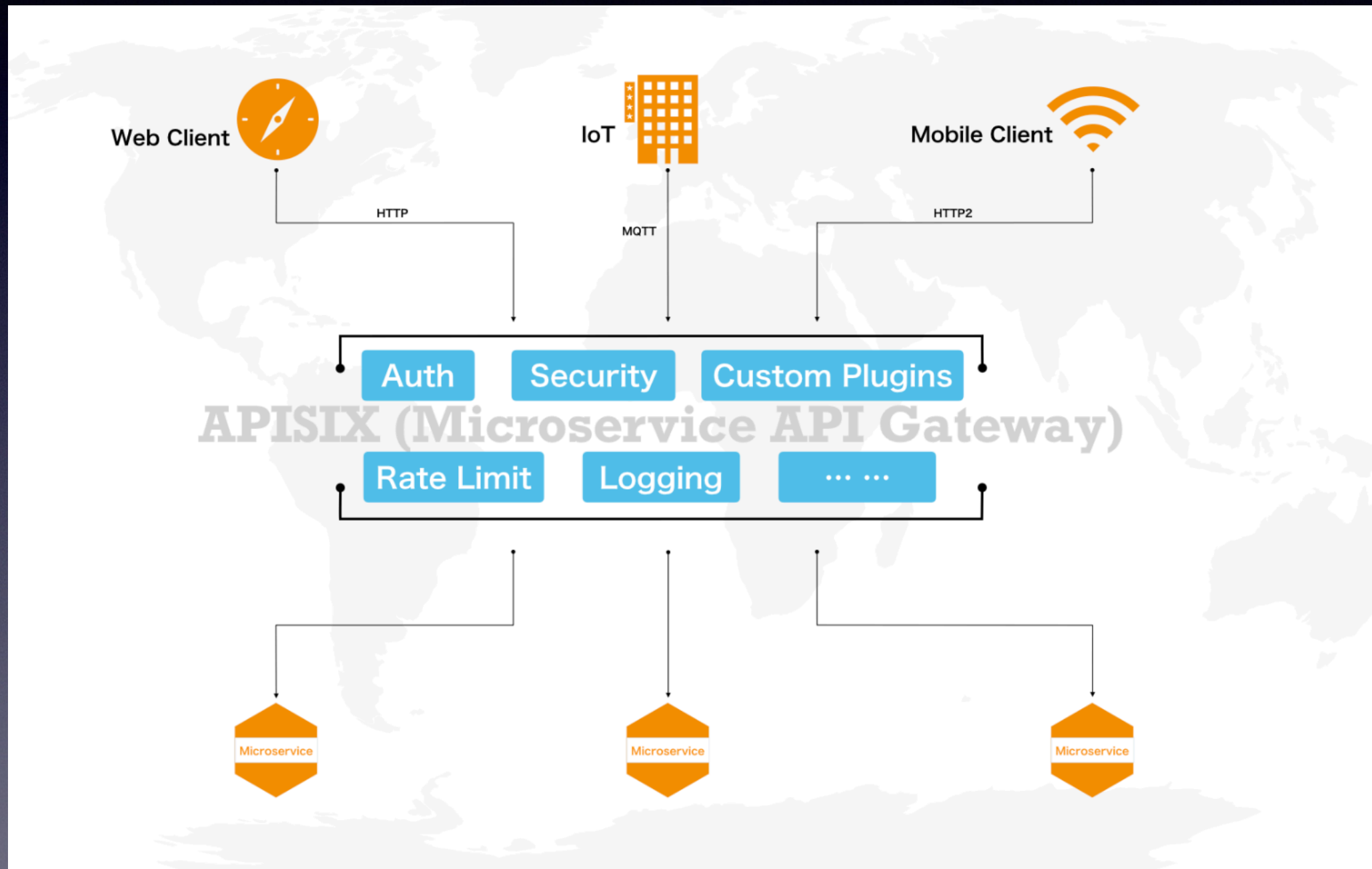
--by Yuansheng

王院生

2014 入职 360 初识
OpenResty, 写了
《OpenResty 最佳实践》
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



什么是 API 网关



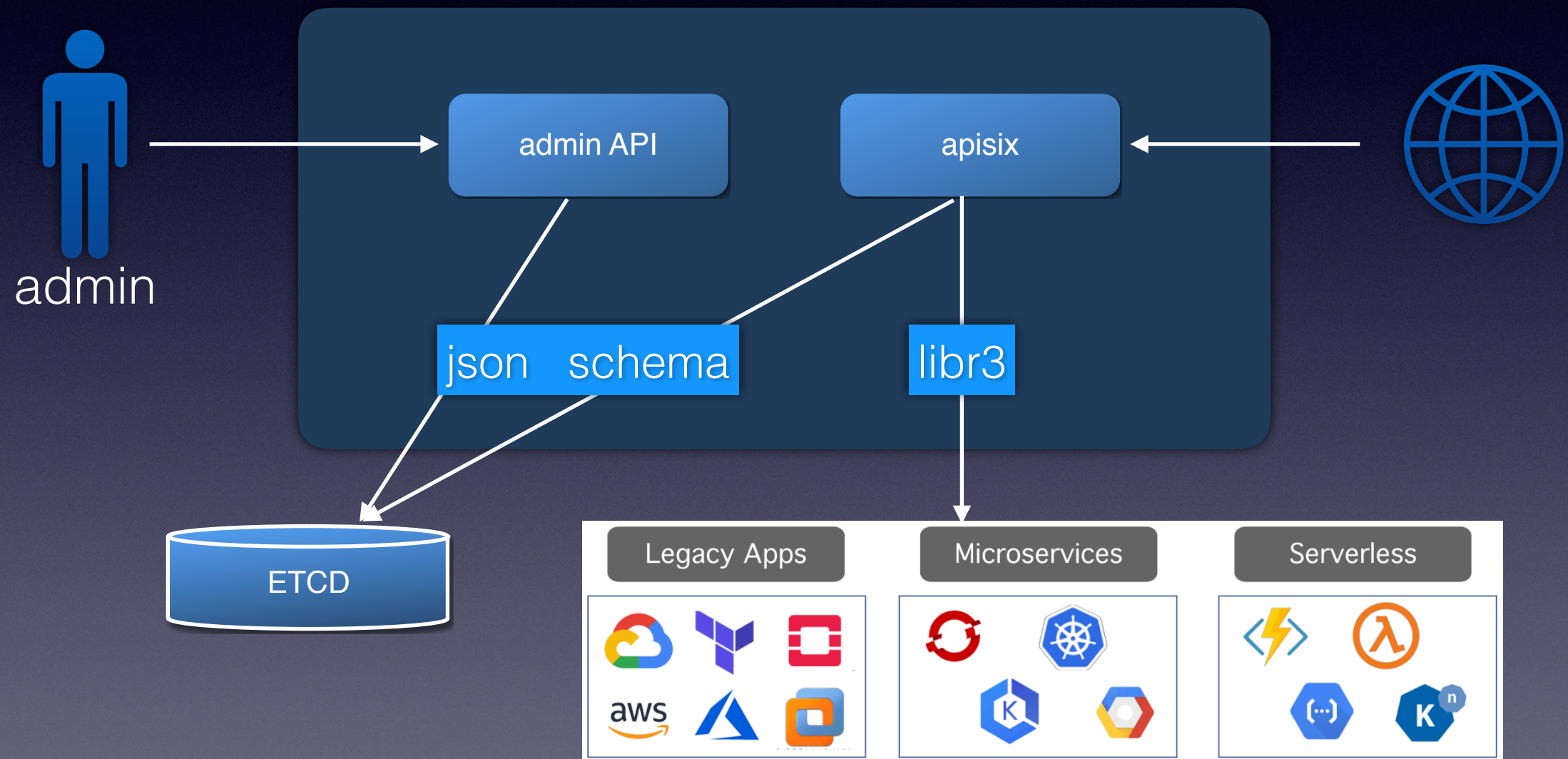
微服务 API 网关

- 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



APISIX 状态

- 目前最新版本 0.5: 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。

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

APISIX 的性能



VS

性能只下降 15%
单 worker: 23-24k 的 QPS
4 worker: 68k 的 QPS
平台: alicloud ecs.ic5.3xlarge

```
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
```


九大技巧助力高性能

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++

九个主要优化技巧

- ngx.log is NYI
- delay_json
- trie + HASH vs scan
- uri + host dispatch
- gc for Lua table

九个主要优化技巧

- 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
- lru_cache 的使用技巧

nginx.log is NYI

delay_json

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


trie + hash vs scan

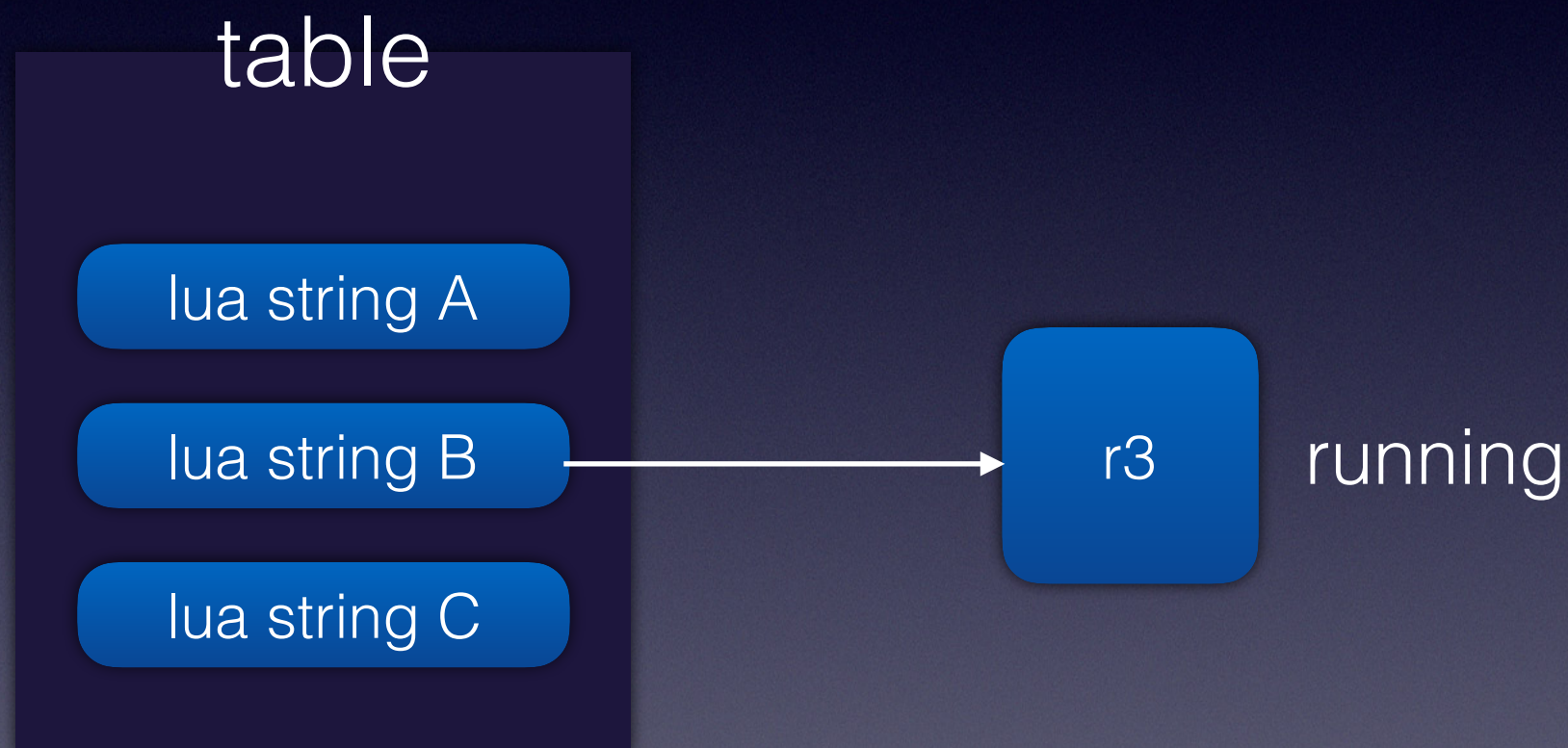
- trie: 借助 libr3 完成前缀等高级匹配, 比如: uri, host
- hash: 完成静态字符串匹配
- scan: 永远都是糟糕的

gc for Lua table

```
-- 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 lru cache 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

性能：apisix 有 3-5% 的提升

```
ngx_int_t
ngx_http_lua_var_fffi_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
- 重用 table
 - 初阶版: `table.clear`
 - 进阶版: `table.pool`

请求之间可以共享 table

```
local function load()  
    ... core.table.clear(local_plugins_hash)  
  
    ... -- ...  
  
    for _, plugin in ipairs(local_plugins) do  
        ... local_plugins_hash[plugin.name] = plugin  
    end  
  
    return local_plugins  
end
```


请求不能共享 table

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


lru-cache 的使用技巧

- lru-cache 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

lru_cache 接口简化

lru_cache 最佳实践用例

```
· 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
```


Q & A

- github.com/iresty/apisix
- 基础库推荐: `apisix/core`
- QQ 交流群: 552030619
- 提供一对一的服务



APISIX: 云原生 API 网关
扫一扫二维码, 加入群聊。