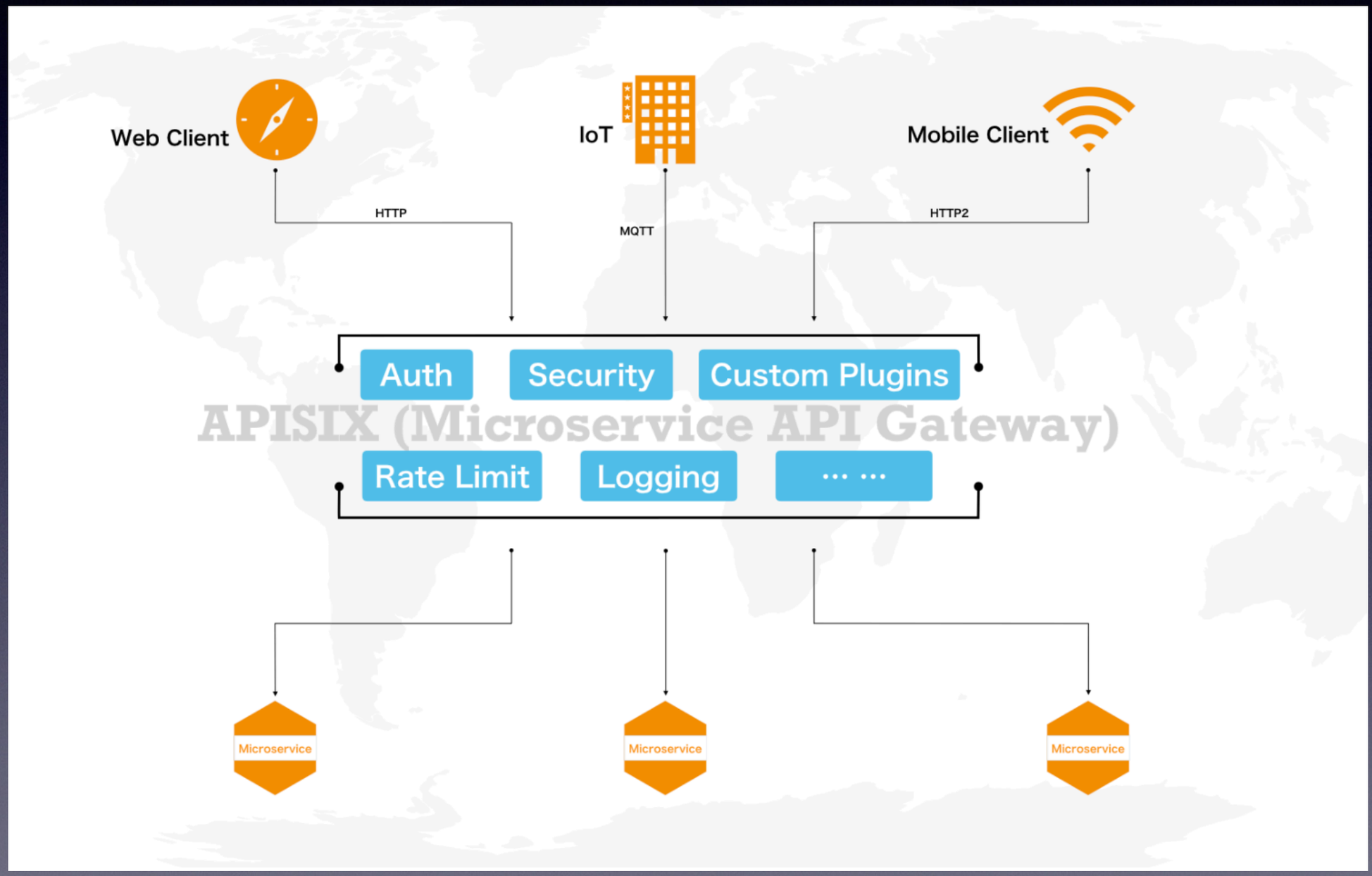
APISIX 高性能实践

--by Yuansheng

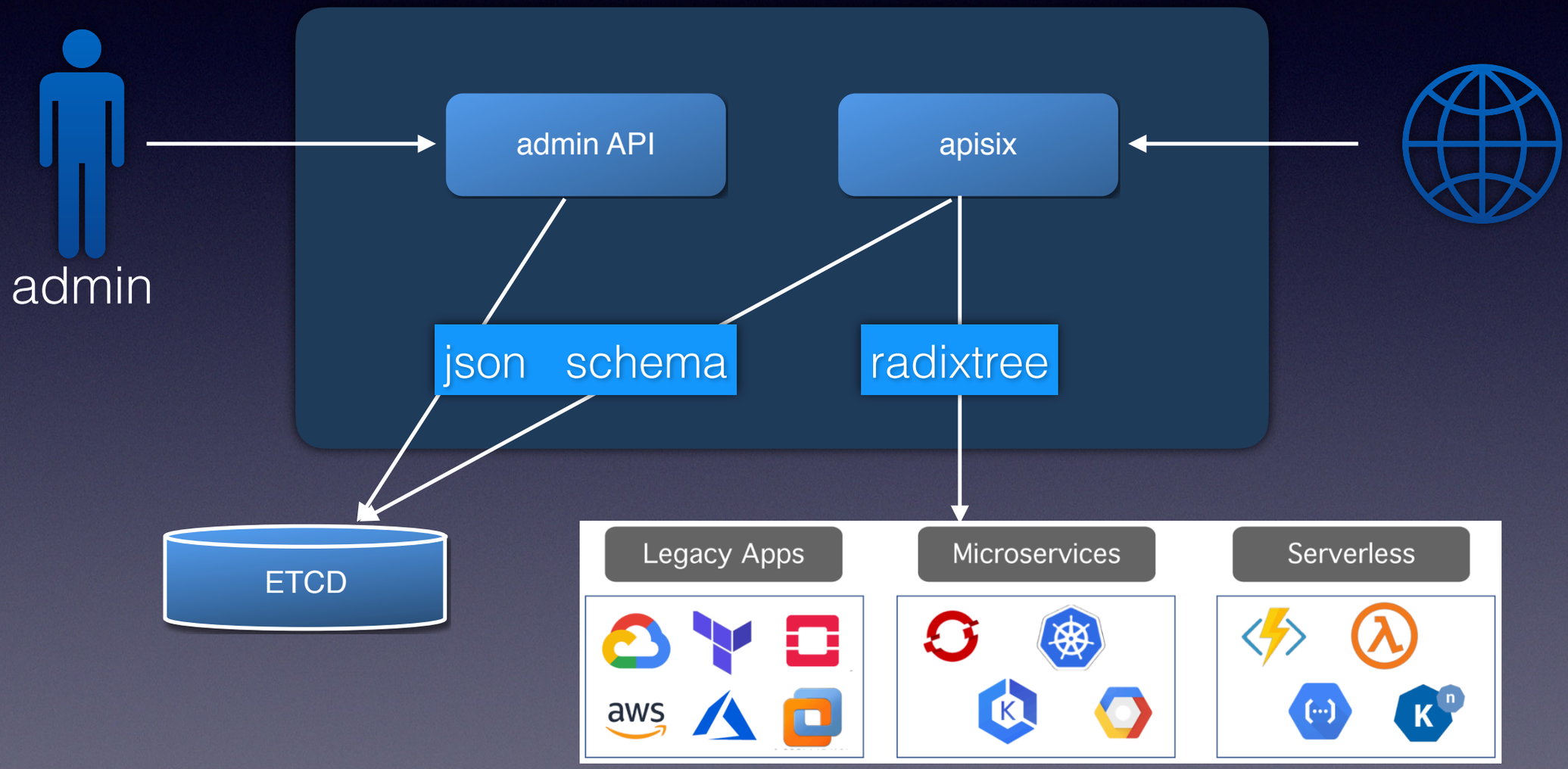
王院生

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


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

APISIX 架构


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

APISIX 状态

- 即将发布最新版本 0.7：
    - etcd + radixtree/libr3 + rapidjson。
    - 超 80% 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 已有功能

- resty-radixtree
- Hot load Plugins
- Health Check
- JWT auth
- OpenTracing
- Serverless
- Dashboard
- Version Control
- Proxy Websocket
- IPv6
- IP whitelist/blacklist
- gRPC transcoding

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

```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
```

why radixtree

- trie tree 比 hash 更快
  - hash 的查找复杂度是 O(K)
  - trie tree 最坏情况的复杂度是 O(k)
  - lua table 的 hash 运算使用 CPU 指令，完美 O(1)
- 匹配模式简单、高效：全量匹配、前缀匹配
- 支持遍历、回调

OpenResty VS Golang

HTTP VS gRPC

gRPC transcode

- HTTP Trailer（patch Test::Nginx)
- 性能比 Golang 的 Fasthttp 略好
- 单核心 QPS 都可以在 10k 左右
- HTTP 与 gRPC 性能相差不大，但 gRPC 体积更小并内置了 schema 检查

ngx.var 的加速

- 5% 的性能提速：
  - iresty/lua-var-nginx-module
- apisix/core/ctx.lua
  - 缓存加速：缓存后百倍加速
  - 缓存更新

fail to json encode

- apisix/core/json.lua
- core.json.encode({…}, true)
  - 强制对 cdata、userdata 等进行编码
  - 有循环嵌套时依然可以打印
- 为日志调试而生：
  - core.json.delay_encode

静态代码检查工具

- apisix：make check

```
$ make check
luacheck -q lua
Total: 0 warnings / 0 errors in 63 files
./utils/lj-releng lua/*.lua lua/apisix/*.lua \
        lua/apisix/admin/*.lua \
        lua/apisix/core/*.lua \
        lua/apisix/http/*.lua \
        lua/apisix/plugins/*.lua \
        lua/apisix/plugins/grpc-transcode/*.lua > \
        /tmp/check.log 2>&1 || (cat /tmp/check.log && exit 1)
```

rapidjson 生命周期

```lua
local function create_validator(schema)
    local sd = schema_doc(schema)
    local validator = schema_validator(sd)

    -- need to cache `validator` and `sd` object at same time
    return {validator, sd}
end

function _M.check(schema, json)
    local validator = cached_sd(schema, nil, create_validator, schema)[1]

    local d = json_doc(json)
    return validator:validate(d)
end
```

第三方库使用 pcre

- resty-libr3 的一个历史 bug
- 跨请求共享对象
- 内部调用了 pcre
- 需要为这个共享对象分配独立内存池

第三方库使用 pcre

```
void *ngx_create_pool(size_t size, void *log);
void ngx_destroy_pool(void *pool);
void *ngx_http_lua_pcre_malloc_init(void *pool);
void ngx_http_lua_pcre_malloc_done(void *old_pool);

extern void *ngx_cycle;

typedef struct {
    void            ****conf_ctx;
    void            *pool;
    void            *log;
} fake_ngx_cycle;

void *memcpy(void *dest, const void *src, size_t n);
]]

local fake_ngx_cycle = ffi.new("fake_ngx_cycle")
C.memcpy(fake_ngx_cycle, C.ngx_cycle, ffi.sizeof("fake_ngx_cycle"))
```

性能得抠

开启 prometheus 插件

APISIX：性能只下降 5%

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