upyun

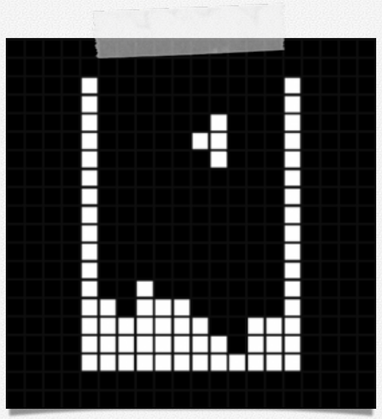
NGINX .conf as a service

♠ UPYUN CDN 2.0 新特性技术概览

Monkey Zhang (timebug)

@ Open Talk No.10

upyun



A Systems Engineer at UPYUN

弓长耳总

★Email: timebug.info@gmail.com

★Github: https://github.com/timebug

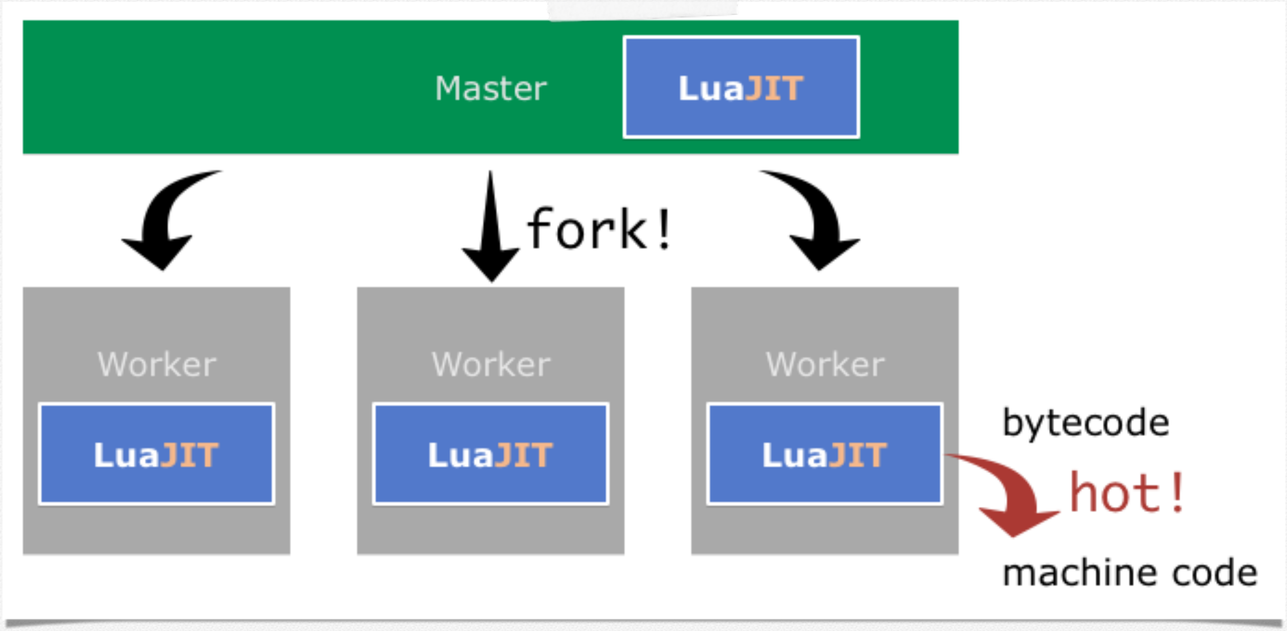
upyun

SSL CDN WAF

```
$ ./configure --prefix=/opt/nginx \
    --add-module=/path/to/lua-nginx-module
```

```
http {
    server {
        listen 8080;

        location /add {
            set $res '';

            rewrite_by_lua '
                local a = tonumber(ngx.var.arg_a) or 0
                local b = tonumber(ngx.var.arg_b) or 0
                ngx.var.res = a + b
            ';

            content_by_lua '
                ngx.say(ngx.var.res)
            ';
        }
    }
}
```

```
$ curl 'http://localhost:8080/add?a=6&b=7'
13
```



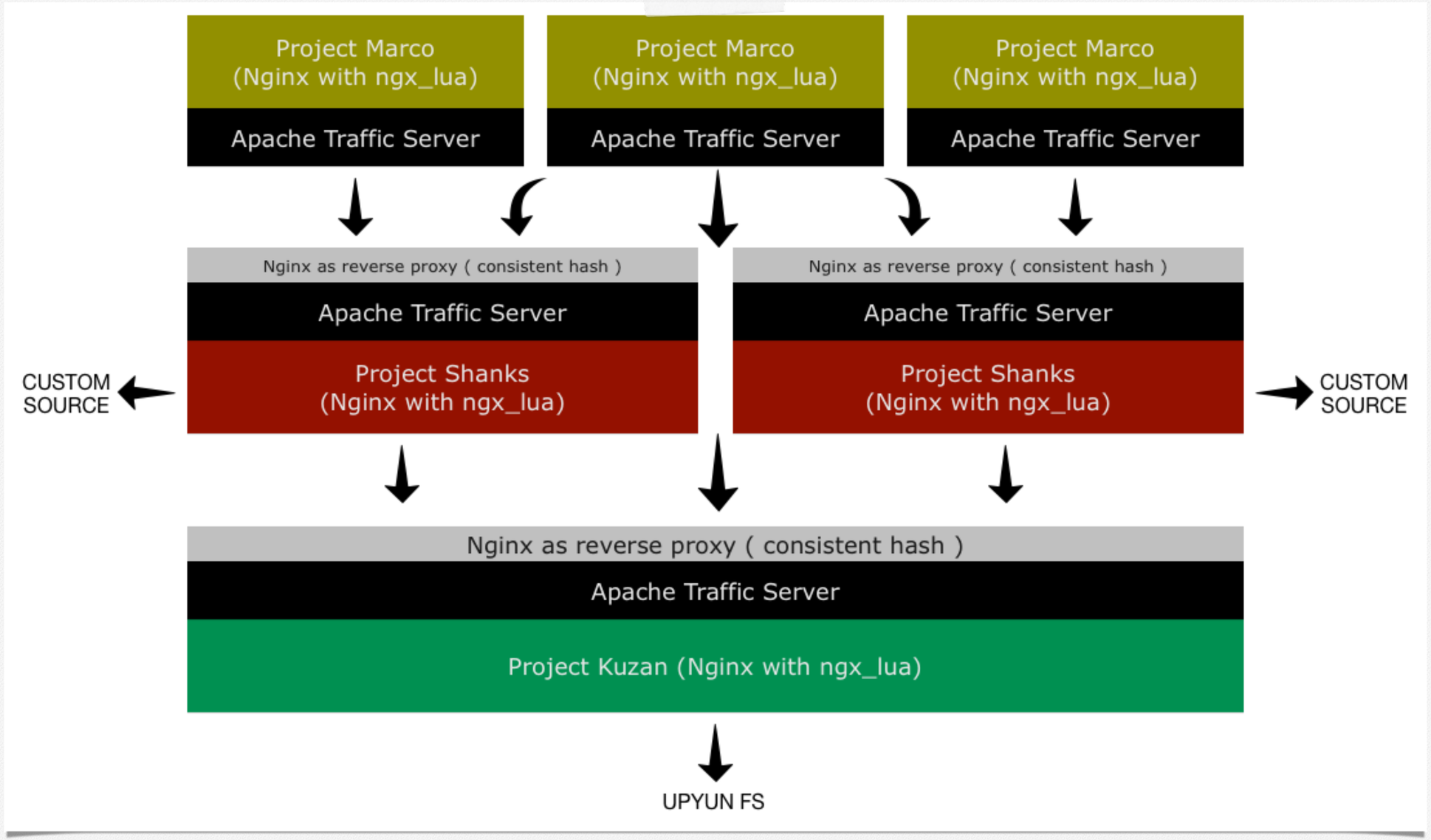
How it works

LuaJIT VM embedded into the `Nginx`

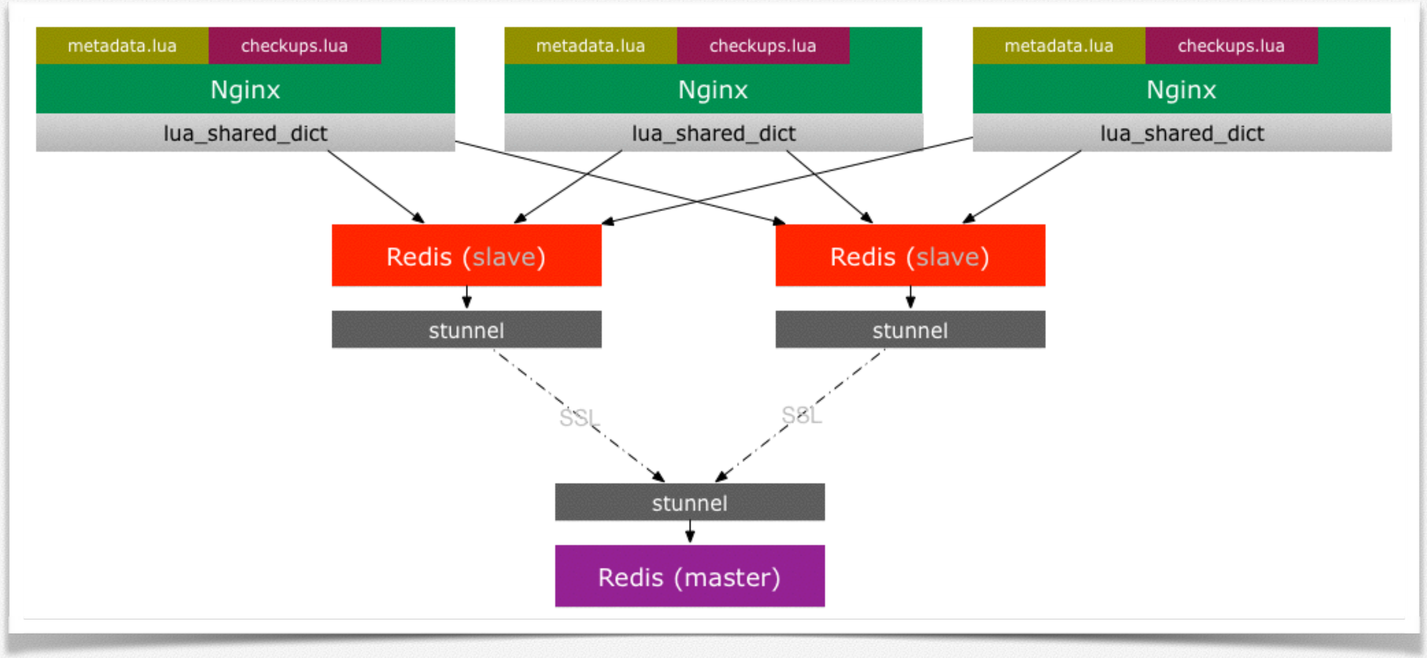
upyun

nginx.conf	service
server_name *.b0.upaiyun.com	Custom Domain Binding
valid_referers, rewrite, **allow, deny**	**Custom Antileech Rules and Redirect:** ip, user-agent, referer, token etc.
expires 7d	**Custom Expires Time:** support specific URI rules etc.
ssl_certificate*	Custom SSL Certificates Load
upstream { server 127.0.0.1 }	Custom CDN Source: support multi-network routing etc.
max_fails=3 fail_timeout=30s **health_check (*)**	Custom Health Check Strategy: passive, active
round-robin, ip_hash, **hash** **(1.7.2+)**	Custom Load Balancing Strategy
…	…

UPYUN CDN 架构


Project Marco
(Nginx with ngx_lua)
Apache Traffic Server
Project Marco
(Nginx with ngx_lua)
Apache Traffic Server
Project Marco
(Nginx with ngx_lua)
Apache Traffic Server
Nginx as reverse proxy (consistent hash)
Apache Traffic Server
Project Shanks
(Nginx with ngx_lua)
Nginx as reverse proxy (consistent hash)
Apache Traffic Server
Project Shanks
(Nginx with ngx_lua)
CUSTOM SOURCE
CUSTOM SOURCE
Nginx as reverse proxy (consistent hash)
Apache Traffic Server
Project Kuzan (Nginx with ngx_lua)
UPYUN FS

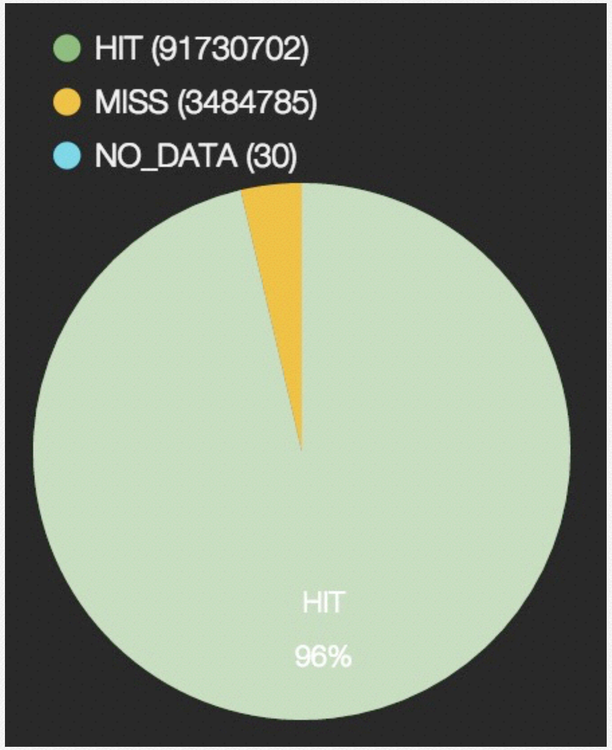
UPYUN CDN 元数据同步：stunnel + Redis

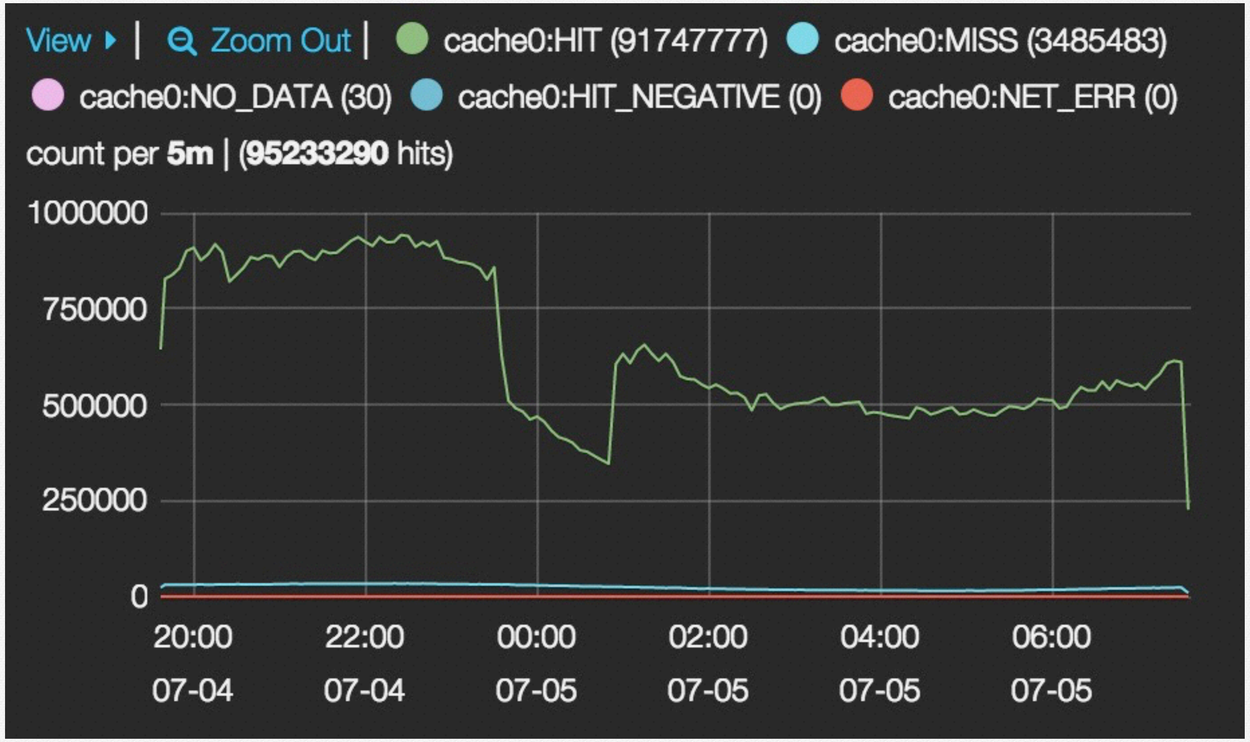


- 每个边缘节点配置了多个从 Redis ，默认只使用其中一个，Marco 会自动进行故障切换
- 同个边缘节点的多个从 Redis 分别主从（数据中心）不同线路的主 Redis
- 基于 Nginx 共享内存对绝大多数元数据进行了 LRU Cache，实际落到从 Redis 的请求并不多


```
“cls:redis": [
    [
        {
            "fail_num": 30921,
            "lastmodified": "2015-07-03 12:07:09",
            "msg": "master link status: down",
            "redis_version": "3.0.0",
            "replication": {
                "master_host": "127.0.0.1",
                "master_last_io_seconds_ago": "-1",
                "master_link_down_since_seconds": "1436051174",
                "master_link_status": "down",
                "master_port": "16379",
                "role": "slave"
            },
            "server": "redis:192.168.11.1:6379",
            "status": "err"
        }
    ],
    [
        {
            "fail_num": 0,
            "lastmodified": "2015-07-03 02:02:38",
            "msg": null,
            "redis_version": "3.0.0",
            "replication": {
                "master_host": "127.0.0.1",
                "master_last_io_seconds_ago": "1",
                "master_link_status": "up",
                "master_port": "16379",
                "role": "slave"
            },
            "server": "redis:192.168.11.2:6379",
            "status": "ok"
        }
    ],
]
```

UPYUN CDN 元数据缓存：命中率高达 95%+





- 缓存状态: **HIT**, **MISS**, **HIT_NEGATIVE**, **NO_DATA**, **NET_ERR**
- 格式转换: JSON (string) -> MessagePack (bytes) -> JSON (string) -> Lua Table

SSL with nginx.conf

upyun

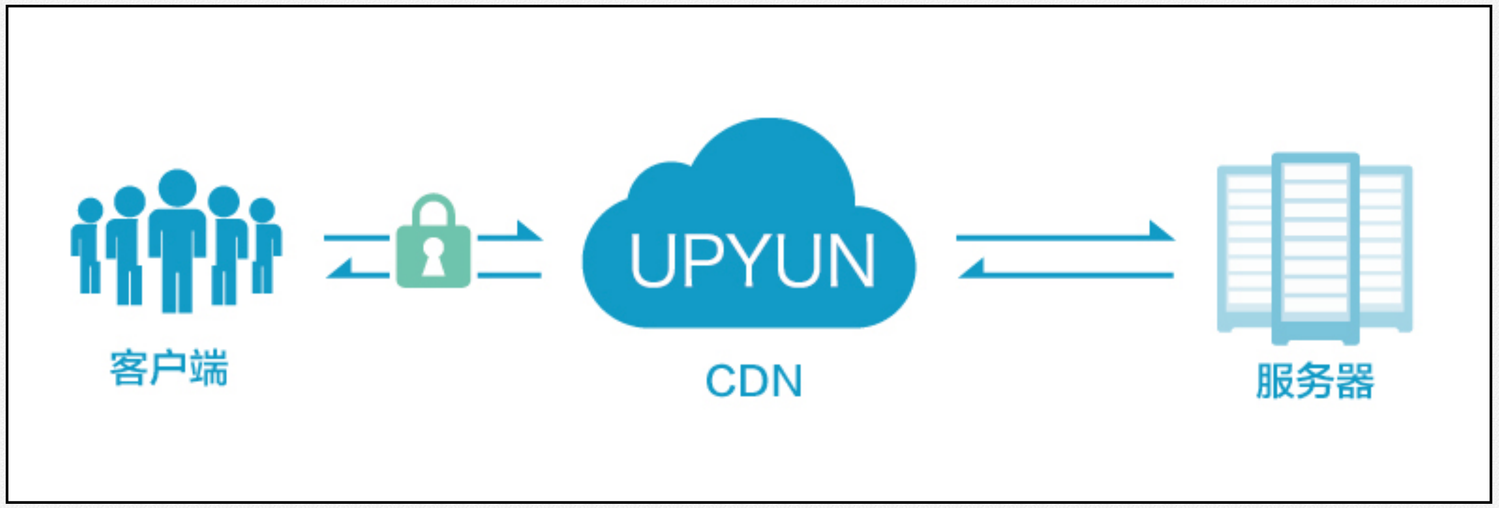
```
server {
    listen 443 ssl spdy;
    server_name upyun.com;

    ssl_certificate       upyun.com.pem;
    ssl_certificate_key   upyun.com.key;

    ssl_ciphers EECDH+CHACHA20:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:EECDH+3DES:RSA+3DES:!MD5;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;

    add_header Strict-Transport-Security max-age=63072000;
    add_header X-Frame-Options SAMEORIGIN;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";
}
```

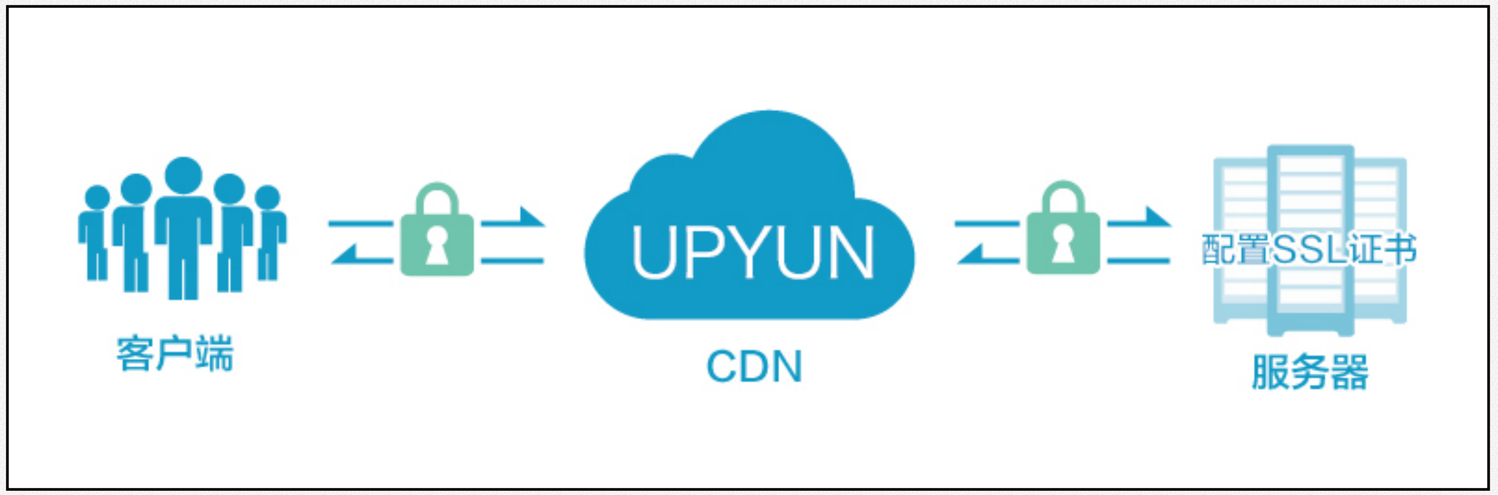
UPYUN 自定义 SSL 服务：基于 SNI 技术


```
{
    "key": "img.hb.aicdn.com",
    "lastmodified": 1392898224,
    "props": {
        "bucket": "hbimg",
        "ssl": {
            "enable": false,
            "force": false,
            "mode": "flexible",
            "crt_id": "ASDFSFSDF",
        }
    }
}
```

```
redis > HSET marco:domains:metadata img.hb.aicdn.com '{"key":"img.hb.aicdn.com","props":{}}'

redis > HSET marco:ssl_crt <crt_id> <证书内容>

redis > HSET marco:ssl_pkey <crt_id> <私钥>
```

UPYUN 自定义 SSL 服务





upsteam with nginx.conf

upyun

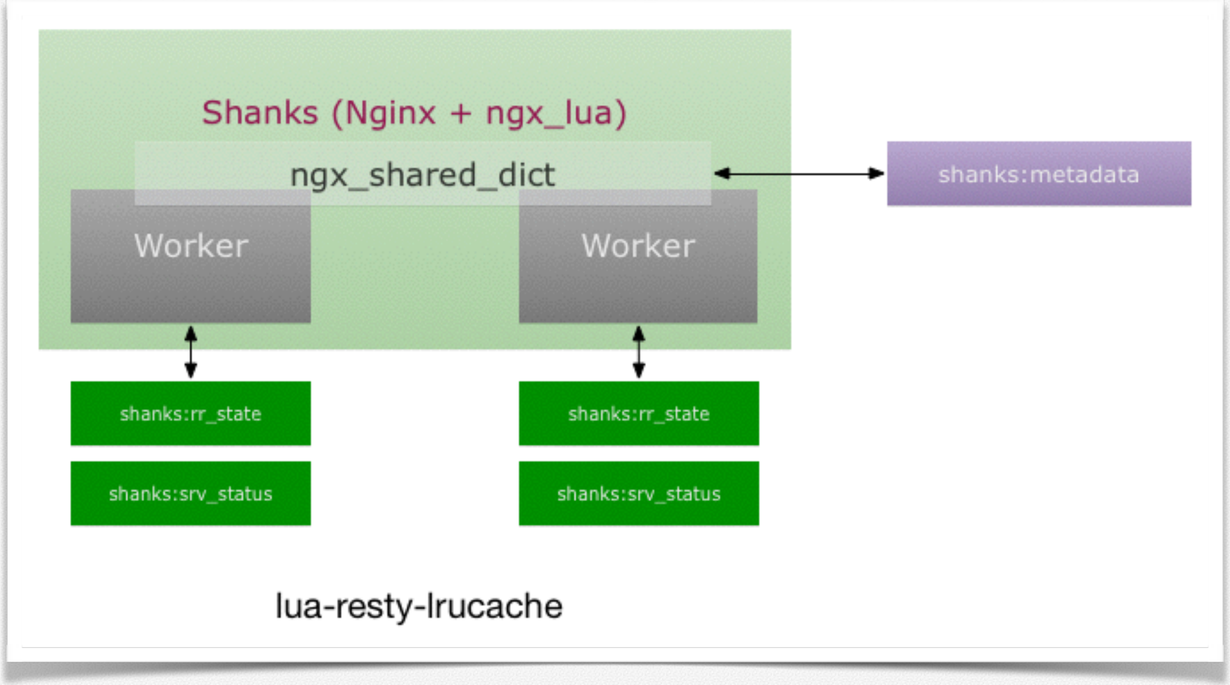
```
upstream blog.upyun.com {
    server 192.168.11.1:8080 weight=1 max_fails=10 fail_timeout=30s;
    server 192.168.11.2:8080 weight=2 max_fails=10 fail_timeout=30s;

    server 192.168.11.3:8080 weight=1 max_fails=10 fail_timeout=30s backup;

    proxy_next_upstream error timeout http_500;
    proxy_next_upstream_tries 2;
}
```

UPYUN 自定义多源站管理

UPYUN 自定义多源站：内部数据结构

```
"cdn": {
    "ctn": {
        "servers": [
            { "host": "122.255.10.11", "port": 80, "weight": 2, "max_fails": 3, "fail_timeout": 30 },
            { "host": "122.255.10.12", "port": 81, "weight": 2, "max_fails": 3, "fail_timeout": 30 },
            { "host": "122.255.10.13", "port": 80, "weight": 5, "max_fails": 3, "fail_timeout": 30 },
            { "host": "122.255.10.14", "port": 81, "weight": 1, "max_fails": 3, "fail_timeout": 30, "backup": true },
            { "host": "122.255.10.15", "port": 80, "weight": 1, "max_fails": 3, "fail_timeout": 30, "backup": true }
        ]
    },
    "cun": {
        "servers": [
            { "host": "182.102.1.12", "port": 8080, "weight": 1, "max_fails": 3, "fail_timeout": 30 }
        ]
    },
}
```

UPYUN 自定义多源站：内部实现



- shanks:metadata 存放回源配置信息（共享内存）
- shanks:rr_state 维护每个空间的轮询状态
- shanks:srv_status 维护每一个回源地址的存活状态
- 回源配置信息全局共享，回源地址状态每个 Worker 独立
- 配合 Redis 元数据同步，全网配置信息实时生效
- 支持同一个线路主备配置，也支持跨线路自动容错

```
local function gen_rr_state_key(bucket_name)
    return str_format("%s:%s", SHANKS_RR_STATE, bucket_name)
end


local function gen_srv_key(net, host, port)
    return str_format("%s:%s:%s:%s", SHANKS_SRV, net, host, port)
end
```

UPYUN WAF 整体设计：规则的生命周期

upyun

- 文本规则
- JSON 规则
- 可执行 Lua 代码
- Redis 分发代码
- 执行代码

UPYUN WAF 整体设计：WAF 执行周期

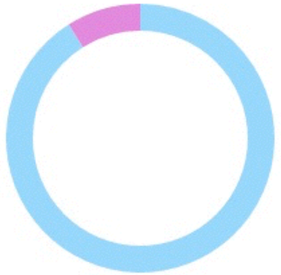
upyun

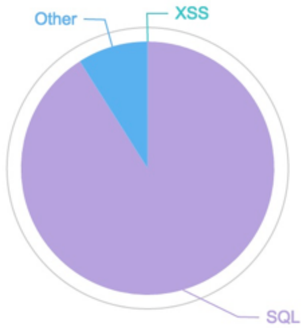
- 设置全局控制变量，例如是否打开 Debug 日志等
- 初始化每个请求的 ctx，用于控制规则的跳转和过滤，比如 ctx.chained 为 true 表示当前处于规则链中
- 获取每个请求的 Collections，包括 Request Headers, Cookies, Args 等，每条规则的 Var 都会涉及到该 Collections 中的某项
- 设置规则代码执行的环境
- 执行每条规则代码，代码执行过程中如果拒绝则以 403 退出，并添加到全局统计变量中

UPYUN WAF 实时攻击信息展示

拦截攻击次数

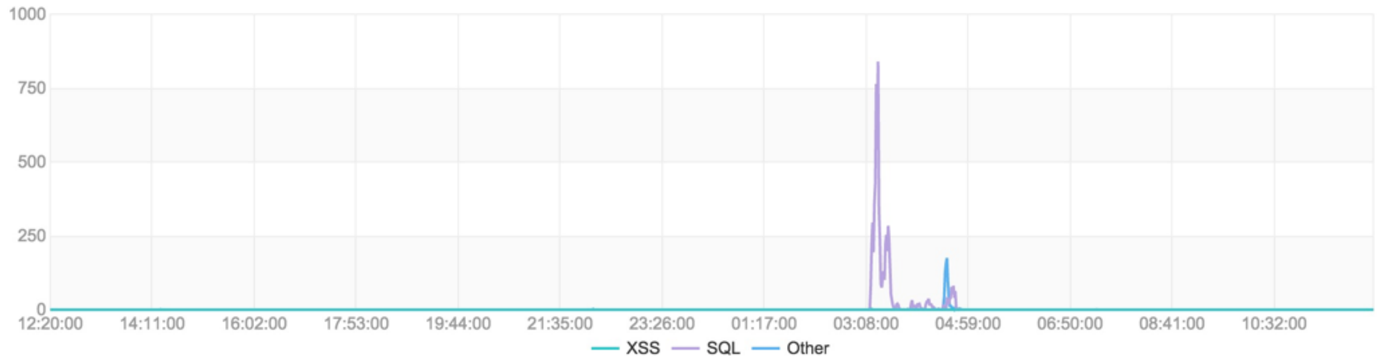
7,975 次



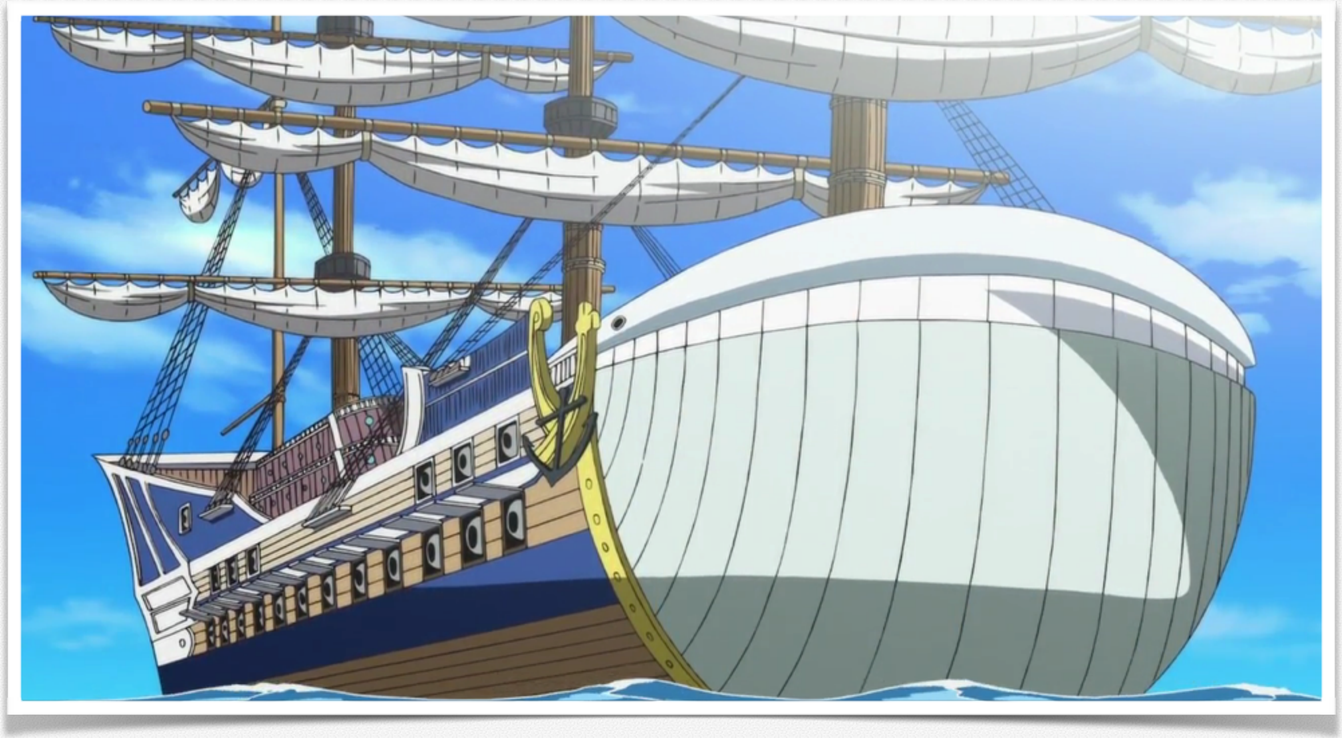


攻击类型	攻击次数
XSS	1
SQL	7,260
Other	717
累积拦截攻击次数	7,978

历史攻击数据



UPYUN CDN 2.0 - Project Moby Dick



UPYUN SSL （自定义 SSL 服务）
UPYUN CDN （动态网站加速服务）
UPYUN WAF （Web 应用防火墙）

- 源站内容动静分离
- 回源支持多线路多回源地址
- 自动缩减 JavaScript, CSS, HTML 文件大小
- JavaScript, CSS 文件支持 Combo 请求
- Maximum Upload Size 最大上传请求大小限制
- 支持域名跟随、参数跟随、协议跟随

Join our team

upyun

©2012-2014 Trybiane

- ★ Github: https://github.com/upyun
- ★ Engineering Blog: http://io.upyun.com/

upyun

Q & A