

Guardian简介

NGINX

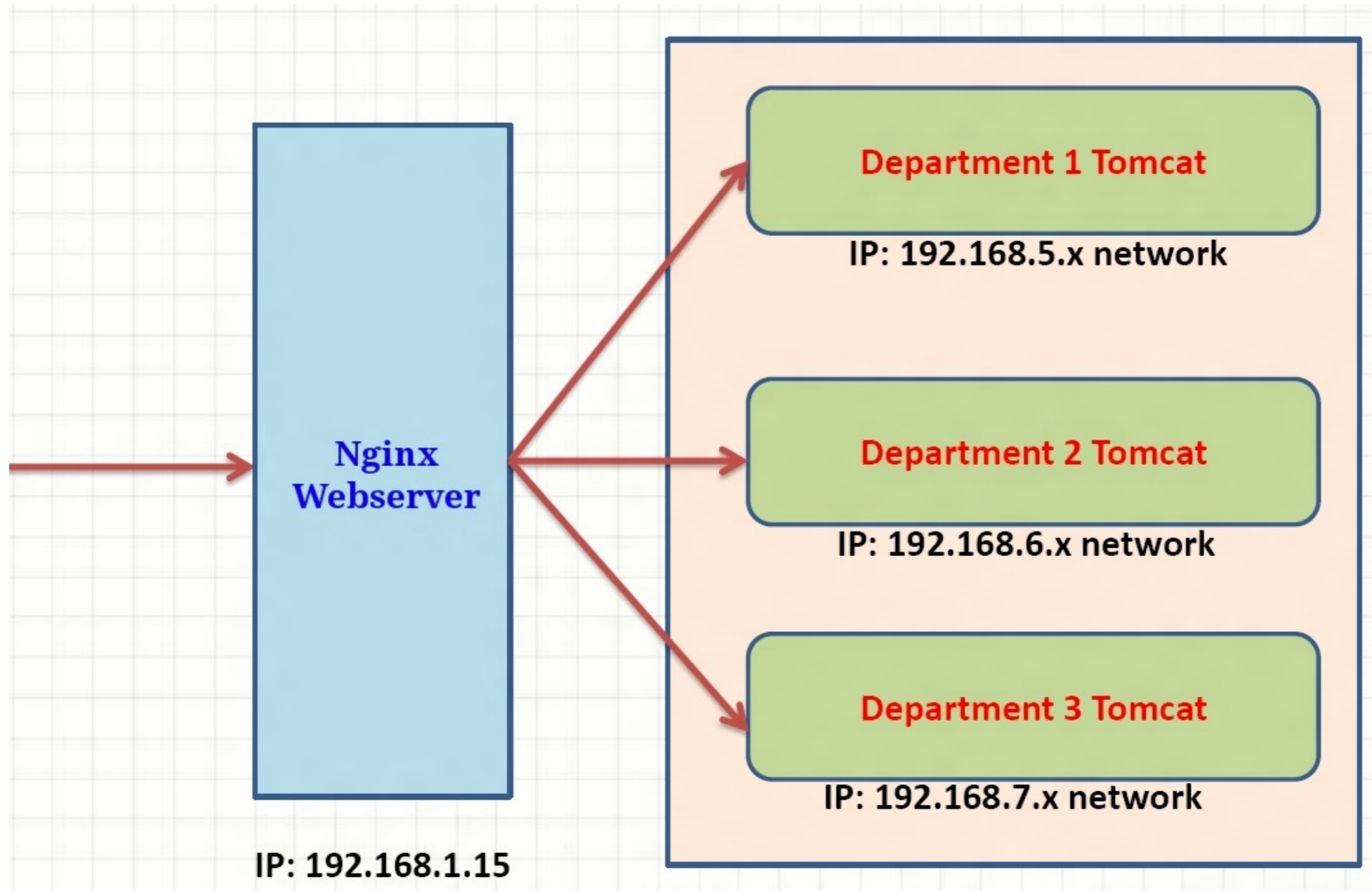
盛宇帆
2016.01



是什么？

做什么？

怎么做？



首先，它是Web Server，将外部请求转发至内网，Nginx有的功能它都有。其次，它能按需转发等Web Server不具有的功能。**Edge Server**

首先，它是一台 **Nginx**

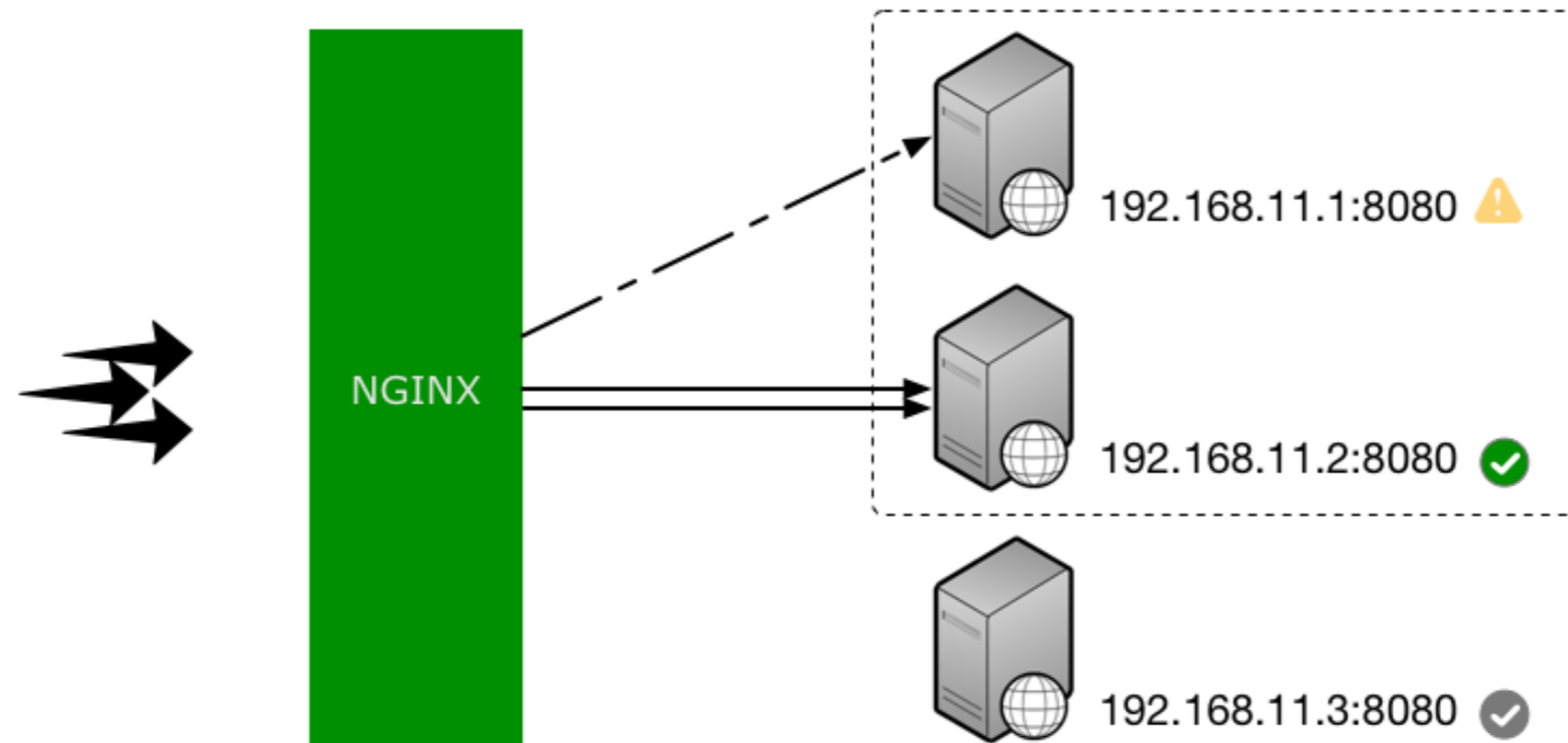
- Static file serving.
- SSL/TLS support.
- Virtual hosts.
- Reverse proxying.
- Load balancing.
- Compression.
- Access controls.
- URL rewriting.
- Custom logging.
- Server-side includes.
- Limited WebDAV.
- FLV streaming.
- FastCGI.



但是，它能做一些Nginx做成不了的事

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

Before



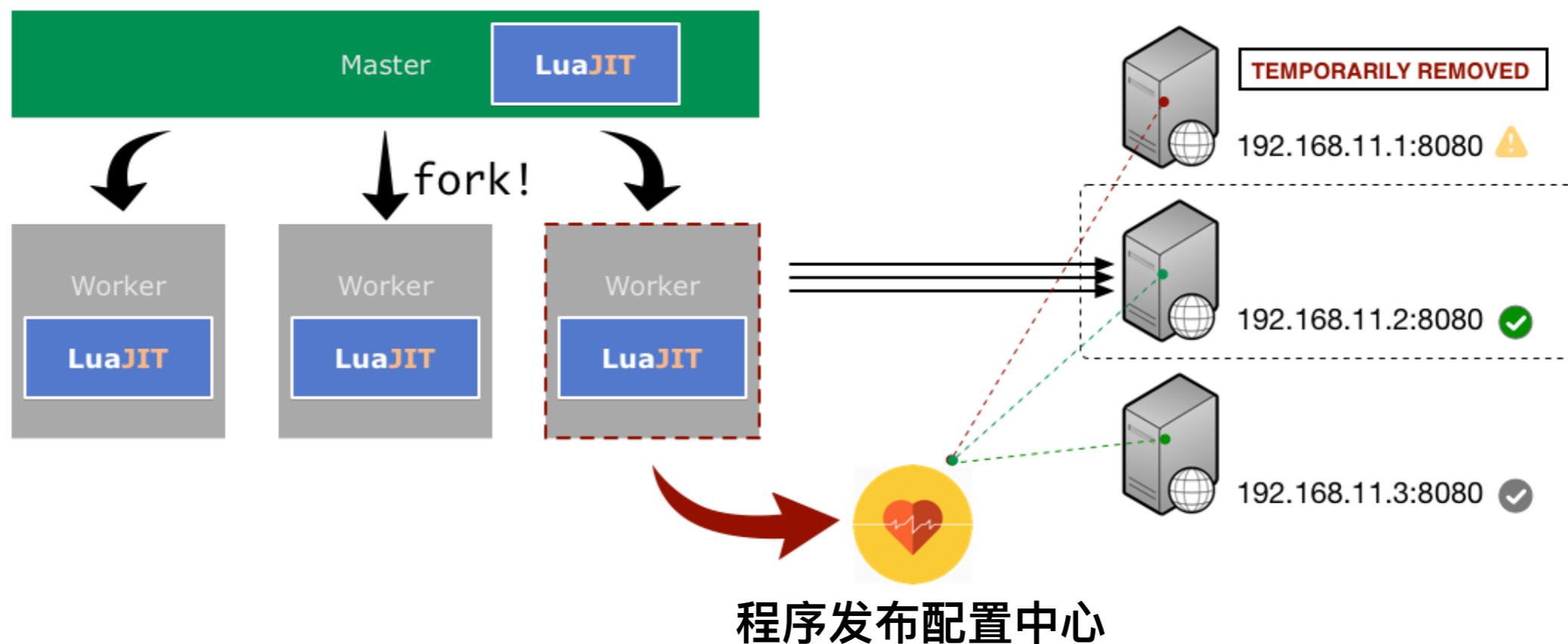
```
upstream ai-dc {  
    server 192.168.11.1:8080 weight=1 max_fails=10 fail_timeout=30s;  
    server 192.168.11.2:8080 weight=2 max_fails=10 fail_timeout=30s;  
  
    server 192.168.11.3:8080 weight=1 max_fails=10 fail_timeout=30s backup;  
  
    proxy_next_upstream error timeout http_500;  
    proxy_next_upstream_tries 2;  
}
```

Guardian 按需分流

```
-- guardian/modules/user_dispatch.lua

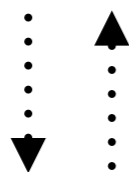
function _M.proxy_backend_server(args_config)
    local args = ngx.req.get_uri_args()
    local license = utils.get_license_from_args(args)
    if license then
        local license_type = cache.rawget(license)
        if not license_type then
            license_type = get_license_type(license, args_config)
            cache.rawset(license, license_type, 3600) -- 3600 seconds, 1 hour
        else
            ngx.log(ngx.INFO, "Get license from shared DICT, license_type= ", license_type)
        end
        if license_type == LICENSE_TYPE.PAYED then
            ngx.log(ngx.INFO, "proxy to paid user, license= ", license)
            if stat_cache then stat_cache:incr("passto_paid_user_count", 1) end
            return "dcpaid"
        end
    end
    ngx.log(ngx.INFO, "proxy to free user, license= ", license)
    if stat_cache then stat_cache:incr("passto_free_user_count", 1) end
    return "dcfree"
end
```

Guardian 灰度发布



```
access_by_lua '  
    local router = require "guardian.modules.dynamtic_router"  
  
    -- 所有 Worker 共享一组配置，实时生效  
    ngx.ctx.backend_server = router.exec_router()  
';
```

Guardian 调用统计



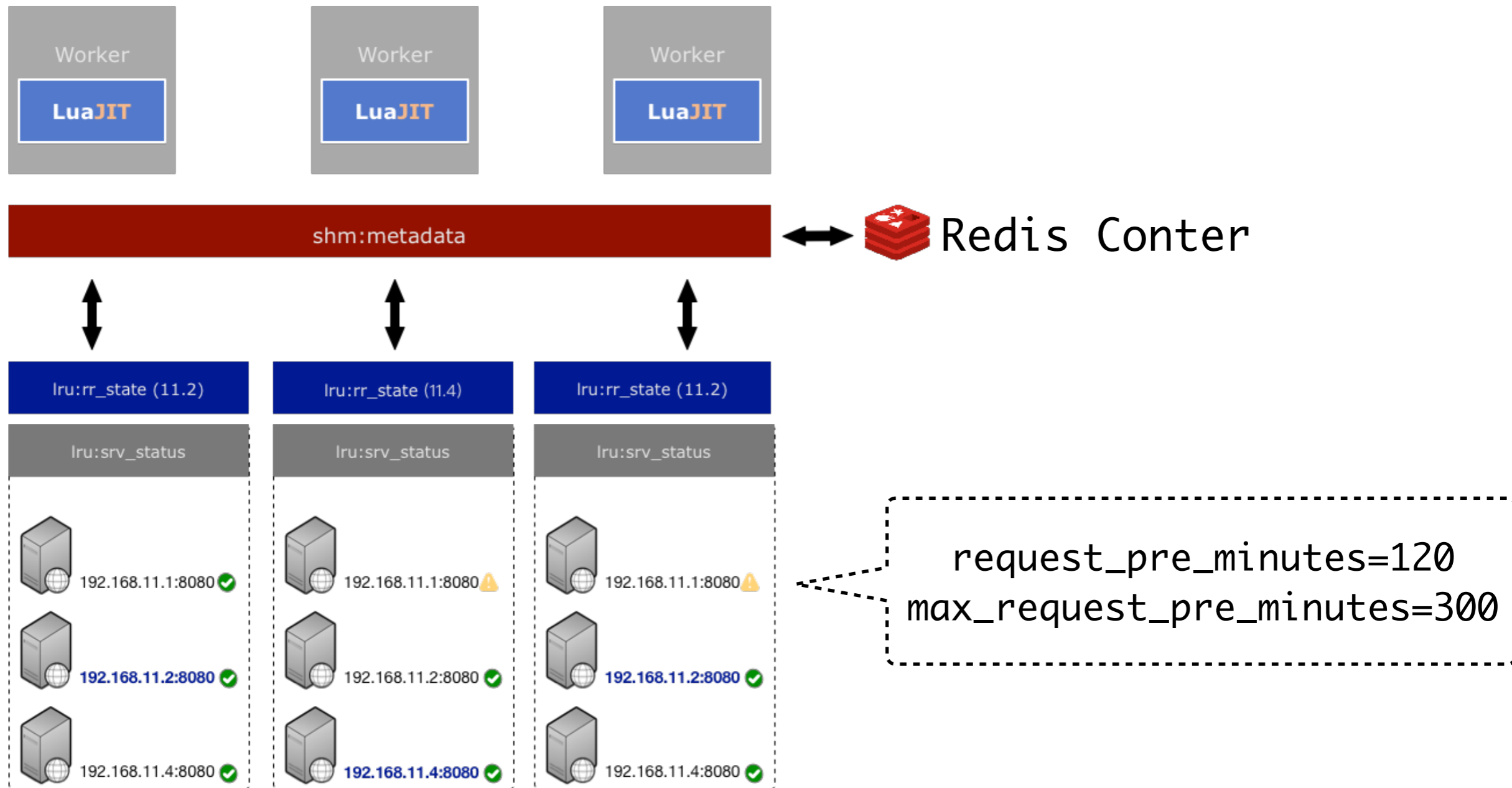
Shm Cache

将请求统计计数缓存，存于Nginx缓存中Redis中，定时缓存至Redis中



Redis

Guardian 频度控制&WAF



NGINX LUA = OpenResty (Guardian)

```
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
```

项目 结构:

~/guardian/nginx_proxy

```
├─ Makefile
├─ README.md
├─ bin
│   └─ guardian
├─ spec
├─ guardian
│   ├─ admin
│   ├─ cli
│   │   ├─ config.lua
│   │   ├─ restart.lua
│   │   ├─ utils
│   │   │   └─ signal.lua
│   │   └─ ssl.lua
│   │   └─ utils.lua
│   │   └─ version.lua
│   ├─ constants.lua
│   ├─ guardian.lua
│   ├─ modules
│   │   ├─ license_operation.lua
│   │   ├─ request_stat.lua
│   │   └─ user_dispatch.lua
│   ├─ tools
│   │   ├─ http_client.lua
│   │   └─ utils.lua
│   └─ vendor
│       └─ ssl.lua
├─ guardian-0.0.1-1.rockspec
└─ guardian.yml
```

.....▶

make install

/usr/local/guardian

```
├─ luajit
├─ nginx_tmp
│   ├─ client_body_temp
│   ├─ fastcgi_temp
│   └─ logs
│       └─ access.log
│           └─ error.log
├─ nginx.conf
├─ proxy_temp
├─ scgi_temp
├─ ssl
│   └─ guardian-default.crt
│       └─ guardian-default.key
└─ uwsgi_temp
```

Quick Start

make deps

make dev

make install

[build/install.sh](#)

```
RUBY_VERSION=2.2.2
LUA_VERSION=5.1.4
LUAJIT_VERSION=2.1.0-beta1
PCRE_VERSION=8.37
LUAROCKS_VERSION=2.2.2
OPENRESTY_VERSION=1.9.3.1
OPENSSL_VERSION=1.0.2e
SERF_VERSION=0.7.0
```

Makefile

```
.PHONY: install dev clean doc lint test coverage uninstall start_test_guardian stop_test_guardian
```

```
install:
```

```
  @if [ `uname` = "Darwin" ]; then \  
    luarocks make guardian-*.rockspec; \  
  else \  
    luarocks make guardian-*.rockspec \  
    PCRE_LIBDIR=`find / -type f -name "libpcre.so*" -print -quit 2> /dev/null | xargs dirname` \  
    OPENSSL_LIBDIR=`find / -type f -name "libssl.so*" -print -quit 2> /dev/null | xargs dirname`; \  
  fi
```

```
dev: install
```

```
  @for rock in $(DEV_ROCKS) ; do \  
    if ! command -v $$rock &> /dev/null ; then \  
      echo $$rock not found, installing via luarocks... ; \  
      luarocks install $$rock ; \  
    else \  
      echo $$rock already installed, skipping ; \  
    fi \  
  done; \  
  bin/guardian config -c guardian.yml -e TEST \  
  bin/guardian config -c guardian.yml -e DEVELOPMENT
```

缓存: 业务现状

- 当前Guardian操作分流的依据的是请求URL中的License Key的参数值，但是需要先从业务系统的数据库中获取License Key对应的用户ID，然后发起HTTP请求用户系统获取付费状态。最后，根据付费状态做流量导向。
- 对于License Key的付费状态，不可能说每次请求去做上述的操作获取更新，那样子会有较大的性能损耗。
- 结合业务特点，对于付费的状态信息，基本上变更的频率并不是那么频繁。所以对于付费状态信息，需要结合License Key做缓存。当前缓存使用OpenResty提供的Shared DICT。缓存策略是对每笔请求，获取License Key，从缓存中去取付费状态。如果缓存中没有，则远程调用获取，获取到了，缓存1个小时。1个小时后缓存失效，则重新调用后端获取付费状态。

缓存：存在问题

1. 初期缓存命中率低的热点问题

应用初期运行的时，因为缓存不存在，所以会出现，大量License key付费状态查询请求。这些请求将会同时落到后端数据库上，可能造成服务器卡顿甚至宕机。请求都会有一定的延迟，由于OpenResty对于阻塞的敏感程度较高，很有可能就会因为超时导致将付费类型变为免费用户的可能。

同时，因为缓存设计的失效时间为1个小时，这个期间，如果链接建立，请求全打在这台机器上，那么基本上更新前无法为付费用户提供高质的服务。

2. 缓存失效策略不严谨问题

如果缓存直接失效，按照一开始的策略，会出现周期性1小时负载问题。也就是说，每过一段时间，缓存就会集体失效，然后并发请求后端服务获取付费状态。如果部署多台，启动间隔较短，那么基本是灾难。

3. 缓存无法实时更新问题

Guardian缓存周期为一个小时，这个期间如果某个用户付费，无法立刻将其的付费状态更新为已经付费。且每个节点的失效时间不一致，N台Guardian节点，理论上全部更新的时间上限可能为N小时。

Lua CDN

Lua WAF

Lua SSL

Lua API

Nginx ngx_lua agentzh **Lua**

blog.cloudflare.com Openresty **LuaJIT** Github

Maxim Dounin Igor Sysoev chaoslawful

<https://groups.google.com/forum/#!forum/OpenResty>

Ansible Michael Pall Open source

Thanks

...