

GOPS

全球运维大会

2019 - AIOps 风向标

GOPS

深圳站

指导单位：



主办单位：



大会时间：2019年4月12日-13日

大会地址：深圳市南山区圣淘沙大酒店（翡翠店）

巧用 Nginx 实现大规模分布式集群的高可用性

陶辉 智链达CTO

目录



1

大规模分布式集群的特点

2

Nginx与scalability

3

Nginx与performance

4

巧用Nginx

internet规模下的分布式环境

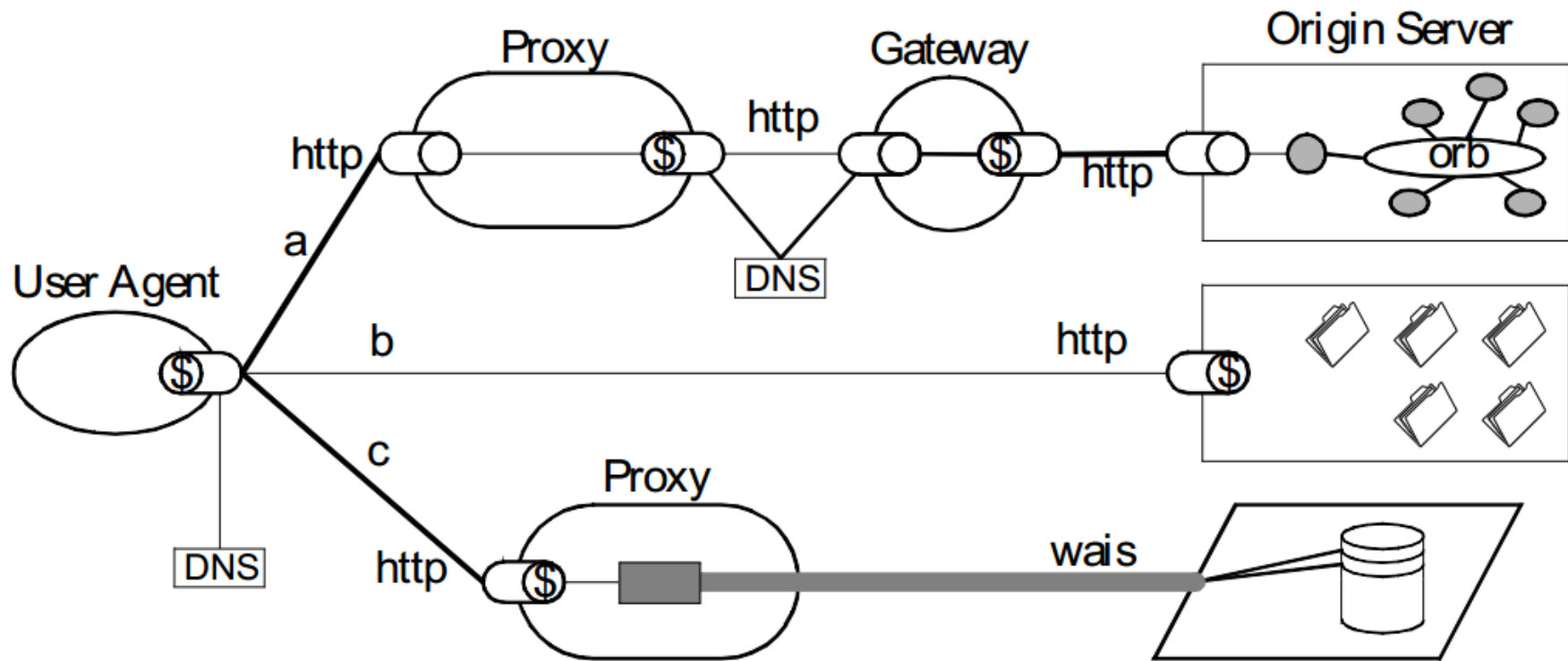
- 多样化的客户端
- 多层代理
- 多级缓存
- 不可控的流量风暴
- 网络安全的高要求
- 快速迭代的业务需求



REST

Representational State Transfer

REST架构下的互联网



架构需求

- 可伸缩性
 - 多种负载均衡策略
- 可扩展性：功能组件的独立进化、新增
- 网络效率
- HTTP协议功能的全面支持：服务器完整覆盖HTTP规范
 - 对HTTP缓存的充分支持

Nginx如何解决internet规模下的分布式网络问题



- 可插拔的模块化设计
 - Uniform Pipe And Filter架构下的丰富生态
 - 多样化的负载均衡策略
 - 基于高效Lua语言的Openresty
- 极致的性能优化
- 对HTTP协议的良好支持
- 优秀的可配置性：支持脚本指令与变量

目录

1 大规模分布式集群的特点



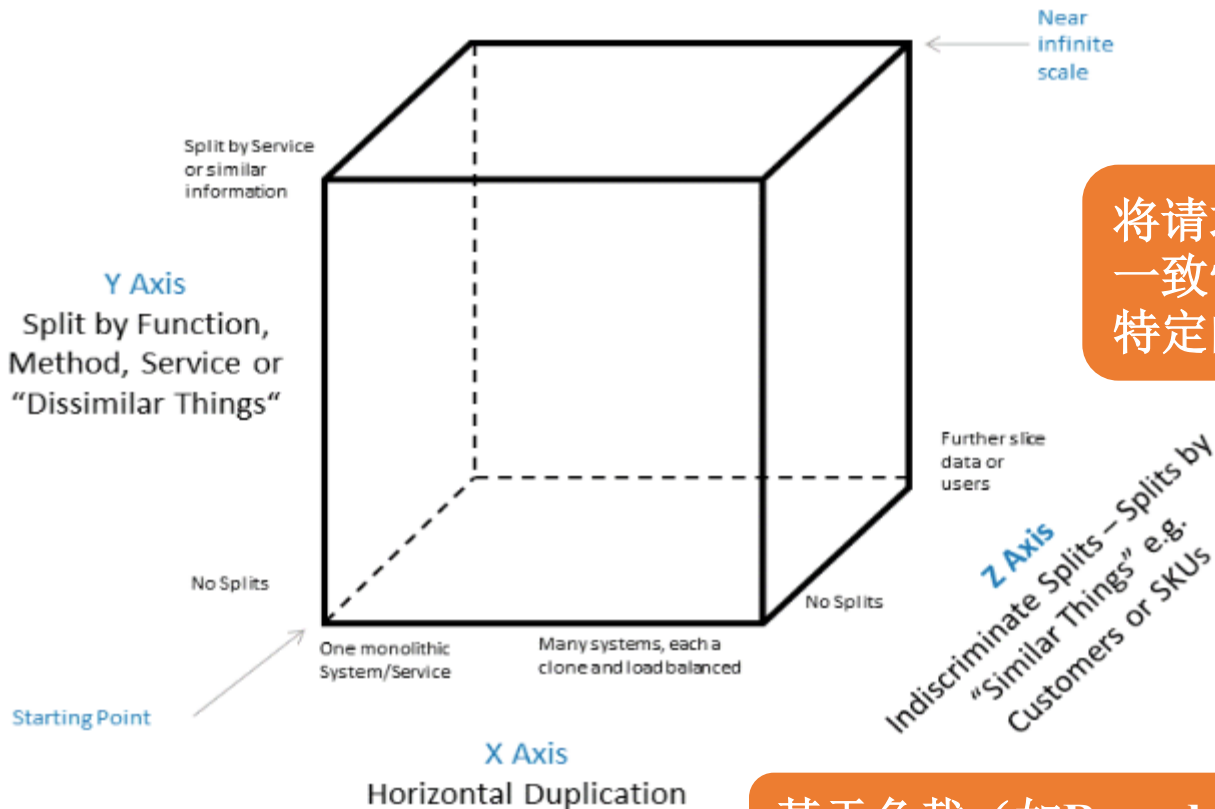
2 Nginx与scalability

3 Nginx与performance

4 巧用Nginx

AKF扩展立方体与Nginx负载均衡

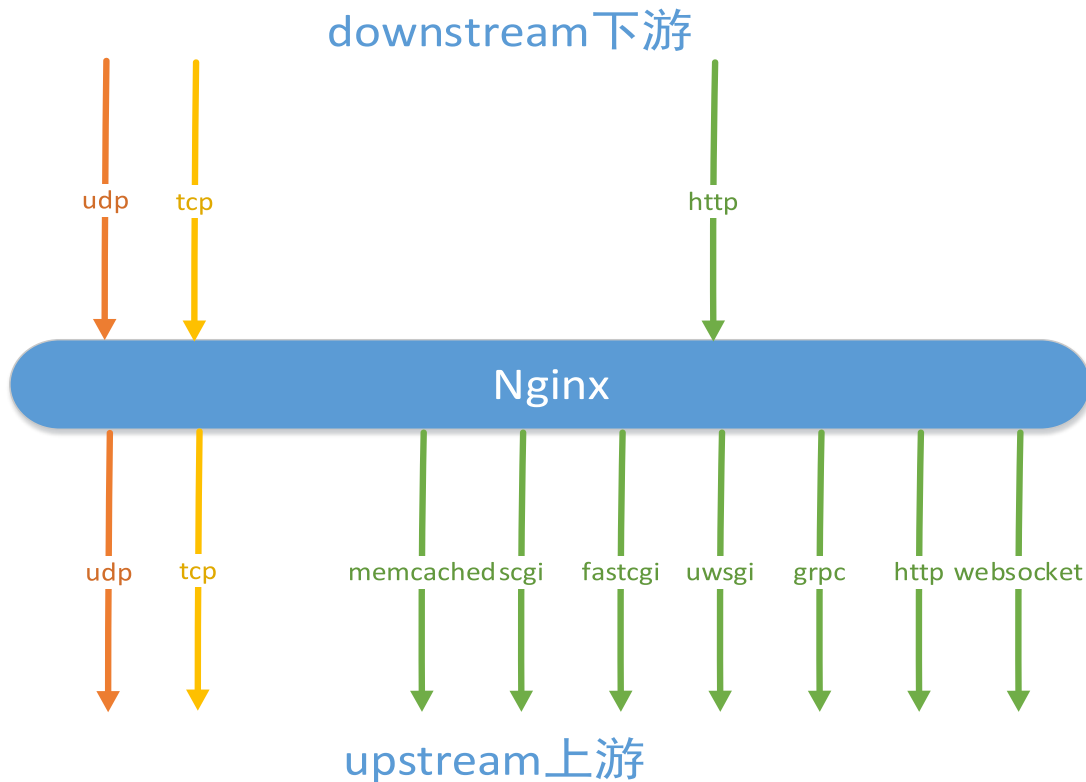
基于资源特征（如URI）对功能进行分发



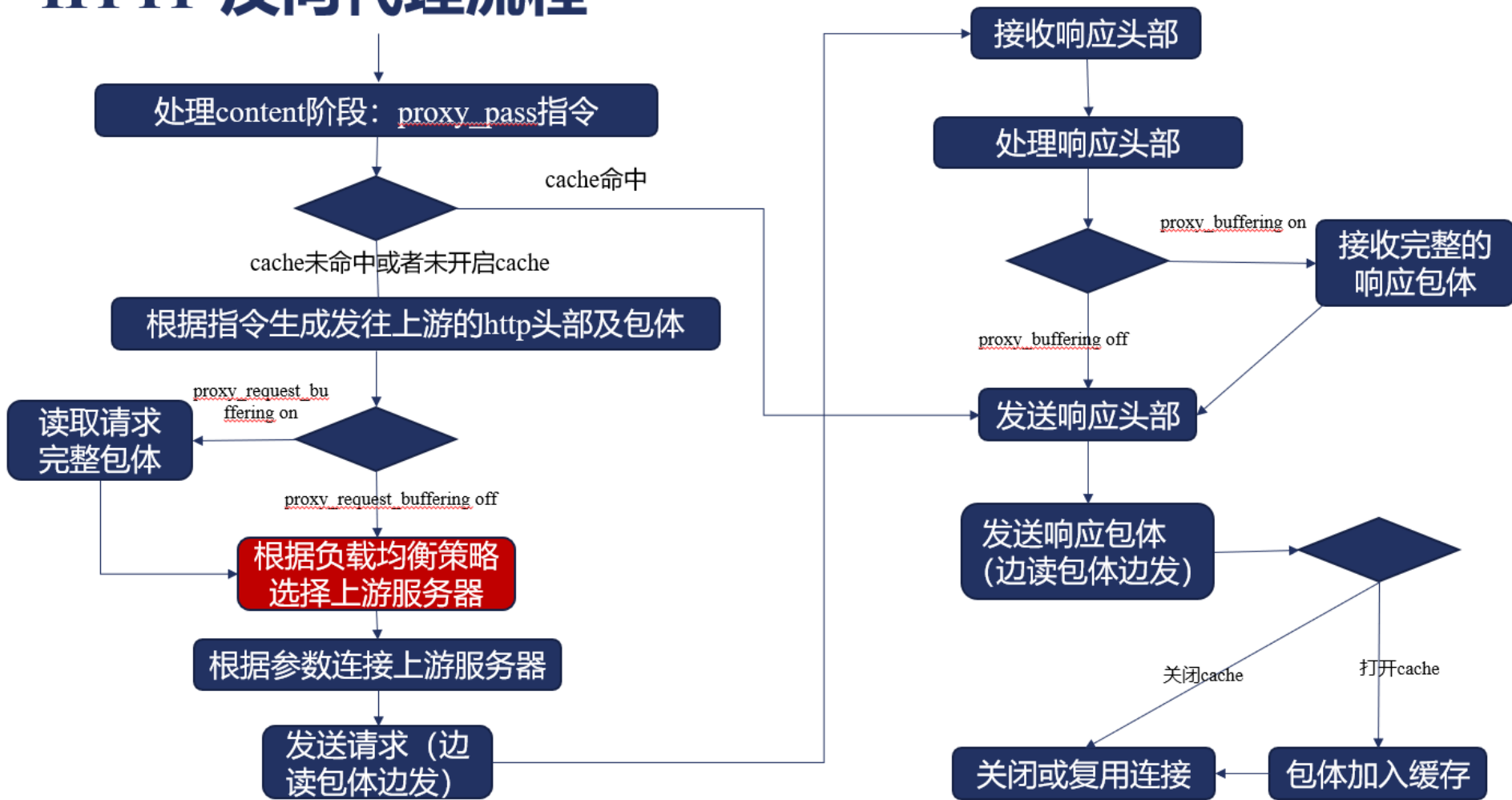
将请求信息映射（如一致性哈希）到某个特定的服务或者集群

基于负载（如Round-Robin或者least-connected算法）分发请求

Nginx支持多种协议转换



HTTP 反向代理流程



Nginx的对外接口：nginx.conf

1. 多级指令配置

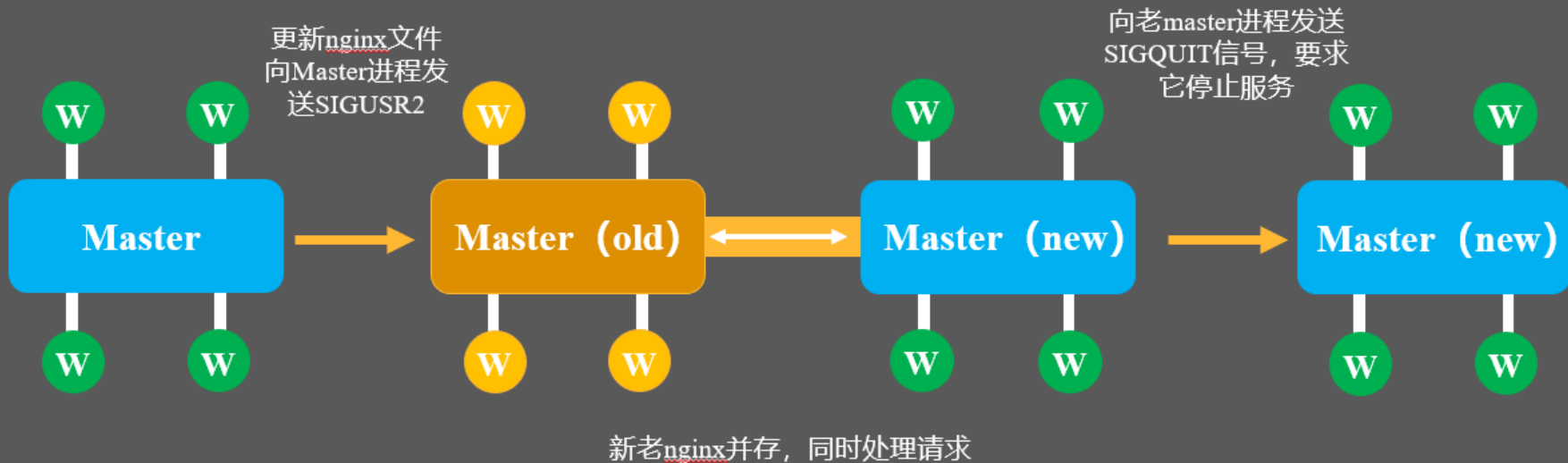
2. 变量

3. 脚本引擎

- if是邪恶的吗？

```
location /only-one-if
{
    set $true 1;
    if ($true) {
        add_header X-First 1;
    }
    if ($true) {
        add_header X-Second 2;
    }
    return 204;
}
```

不停机更新nginx二进制文件



目录

1 大规模分布式集群的特点

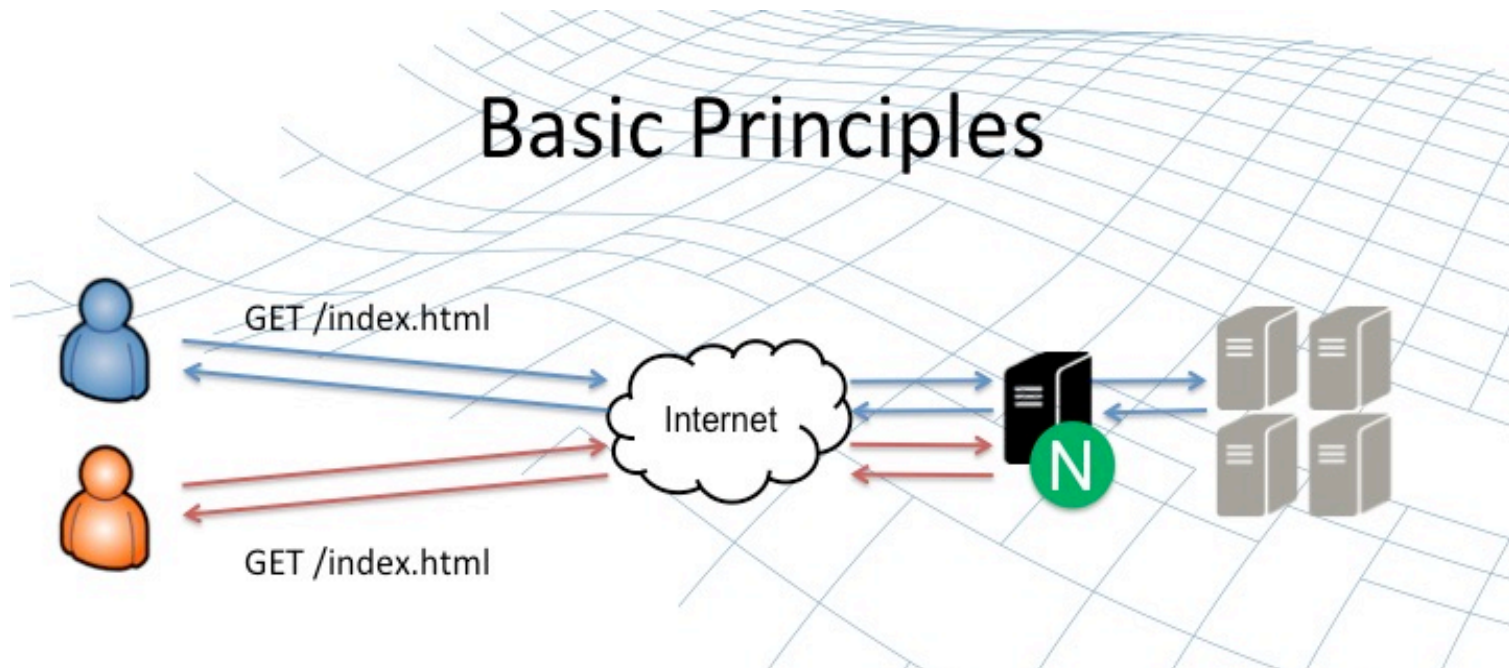
2 Nginx与集群scalability

➔ **3** Nginx与集群performance

4 巧用Nginx

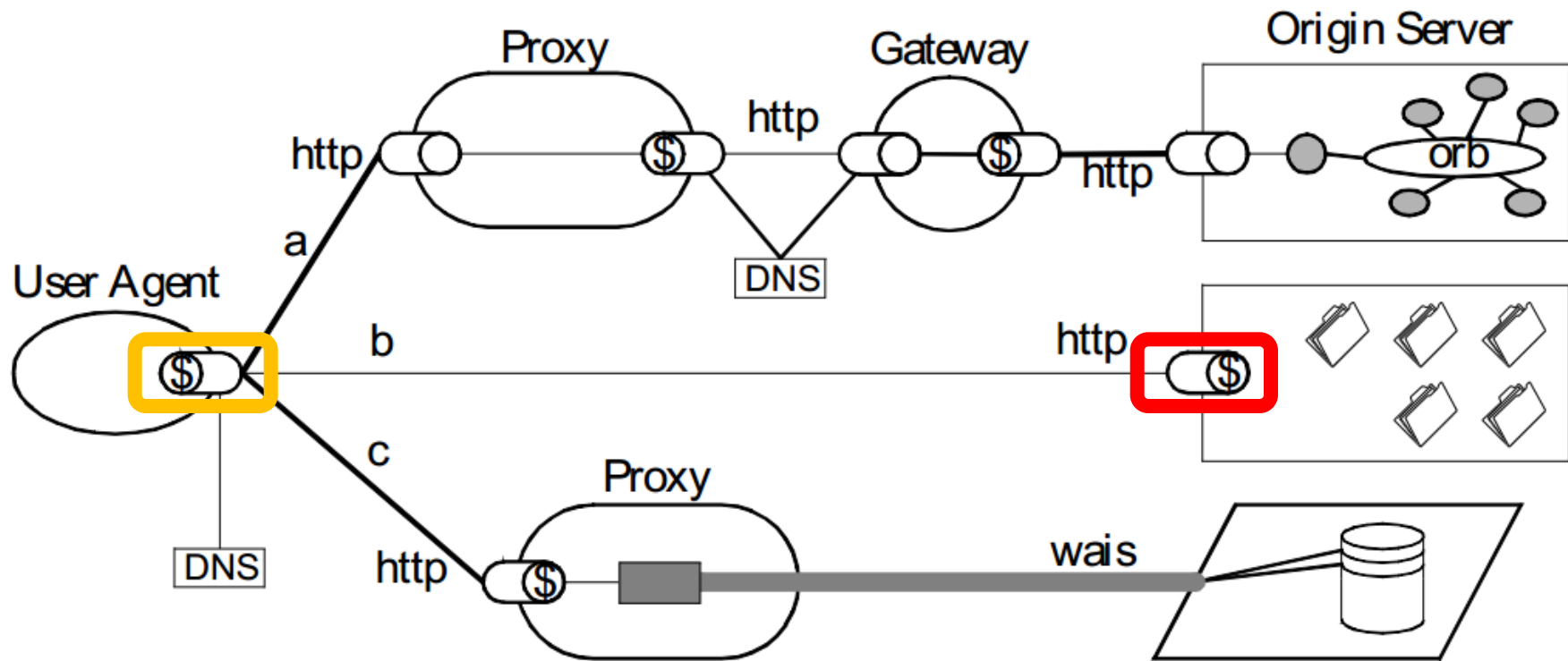
REST架构下的共享缓存

Basic Principles



Used by: Browser Cache, Content Delivery Network *and/or* Reverse Proxy Cache

共享缓存与私有缓存

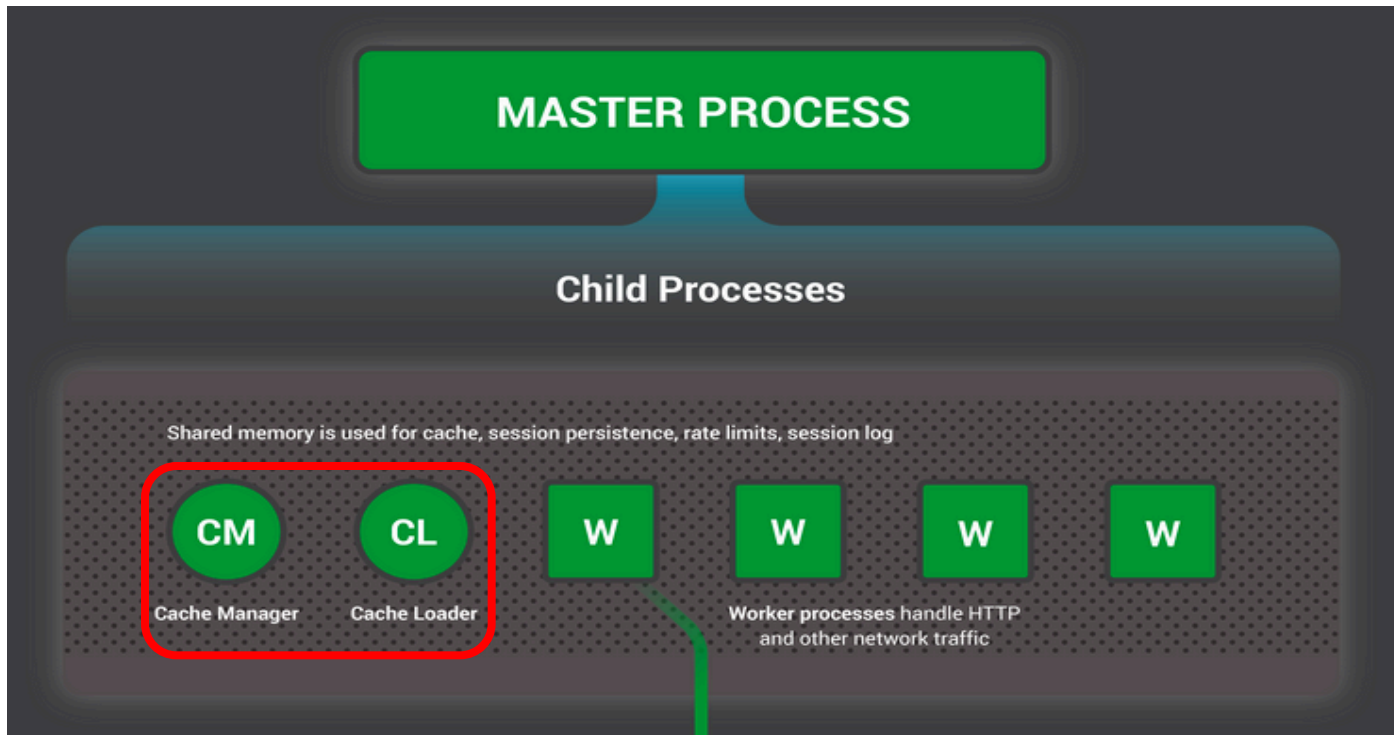


REST架构下的缓存

- HTTP HEADER

- Vary
- Cache-Control
 - 请求：max-age,max-stale,min-fresh,no-cache,no-store,no-transform,only-if-cached
 - 响应：must-revalidated,no-cache,no-store,no-transform,public,private,proxy-revalidated,max-age,s-maxage
- Age
- Warning
- Set-Cookie
- Expires
- Pragma
- Last-Modified

Cache Loader与Cache Manager进程



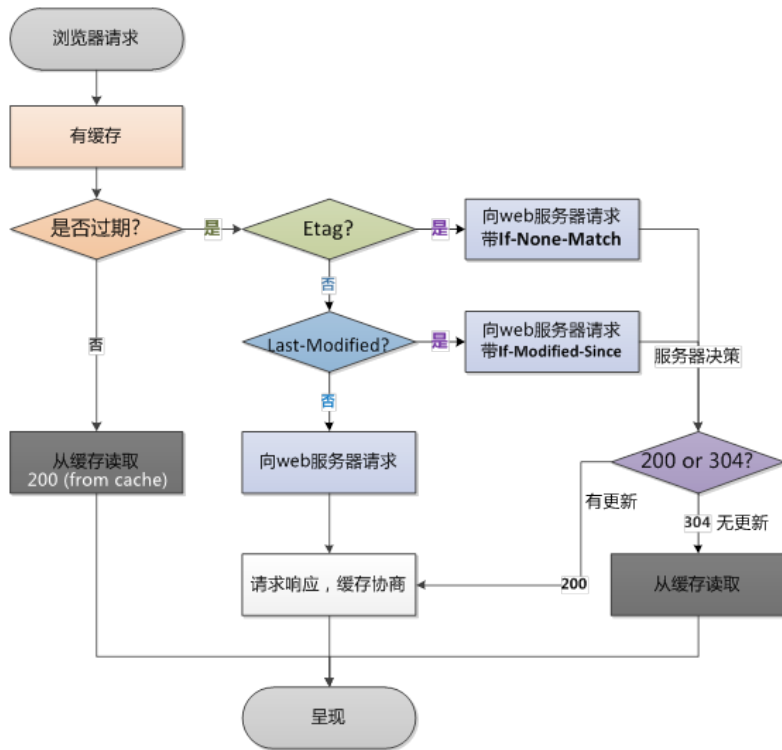
有条件的更新缓存

• HTTP HEADER

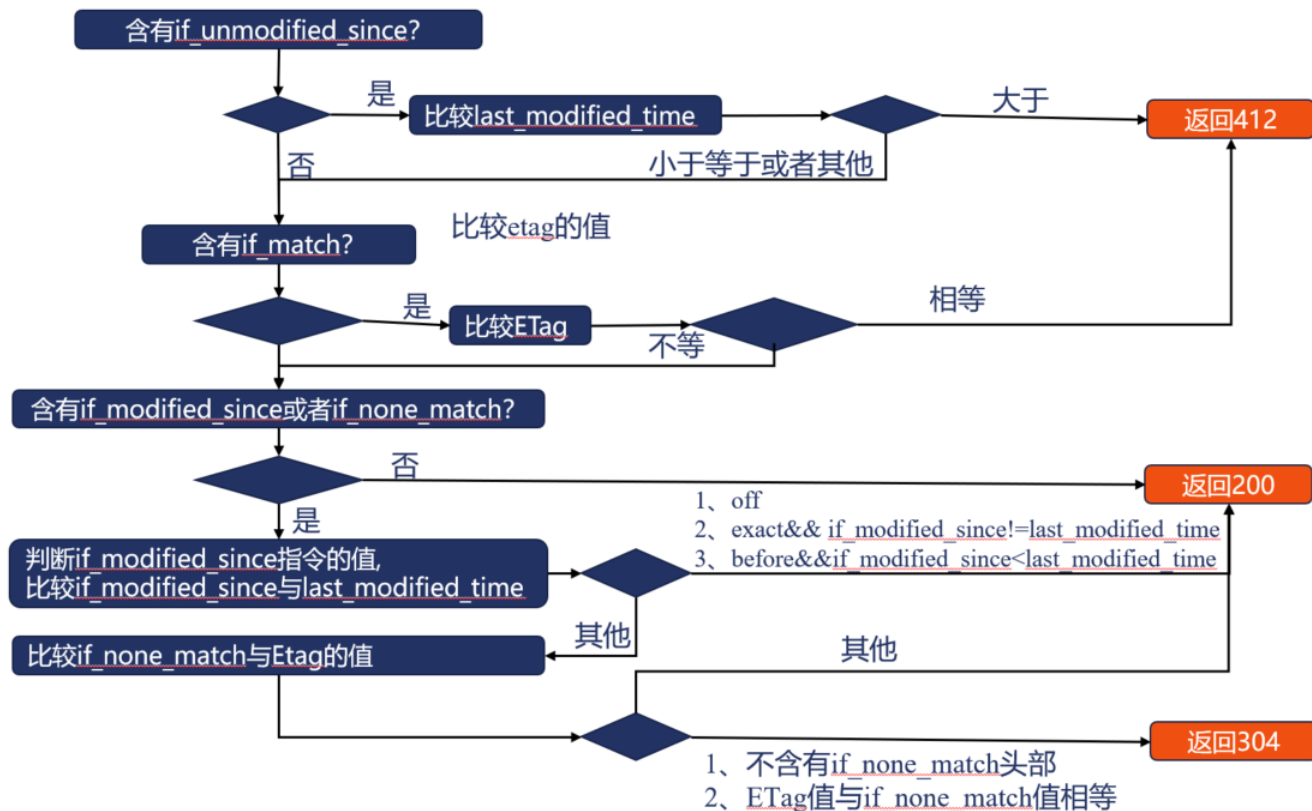
- Etag
- If-Match
- If-Modified-Since
- If-None-Match
- If-Unmodified-Since
- Last-Modified

• 返回状态码

- 304
- 412



ngx_http_not_modified_filter_module模块



合并回源请求

减轻峰值流量下的压力

Syntax: **proxy_cache_lock** **on** | off;

Default: proxy_cache_lock off;

Context: http, server, location

Syntax: **proxy_cache_lock_timeout** *time*;

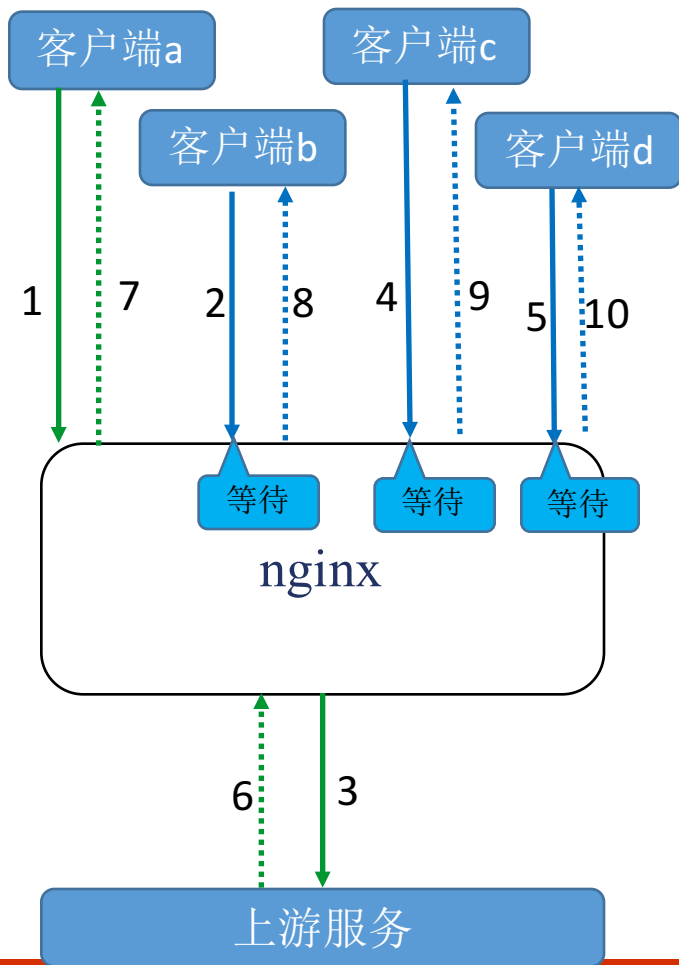
Default: proxy_cache_lock_timeout 5s;

Context: http, server, location

Syntax: **proxy_cache_lock_age** *time*;

Default: proxy_cache_lock_age 5s;

Context: http, server, location



减少回源请求 使用stale陈旧的缓存

Syntax: `proxy_cache_use_stale error | timeout |
invalid_header | updating | http_500 | http_502 |
http_503 | http_504 | http_403 | http_404 | http_429
| off ...;`

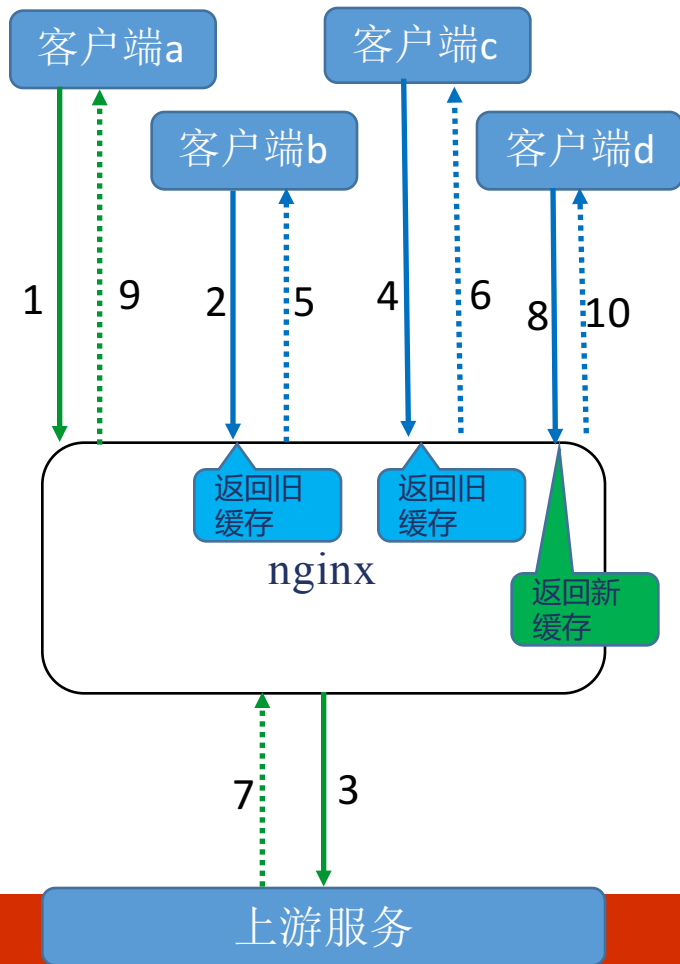
Default: `proxy_cache_use_stale off;`

Context: `http, server, location`

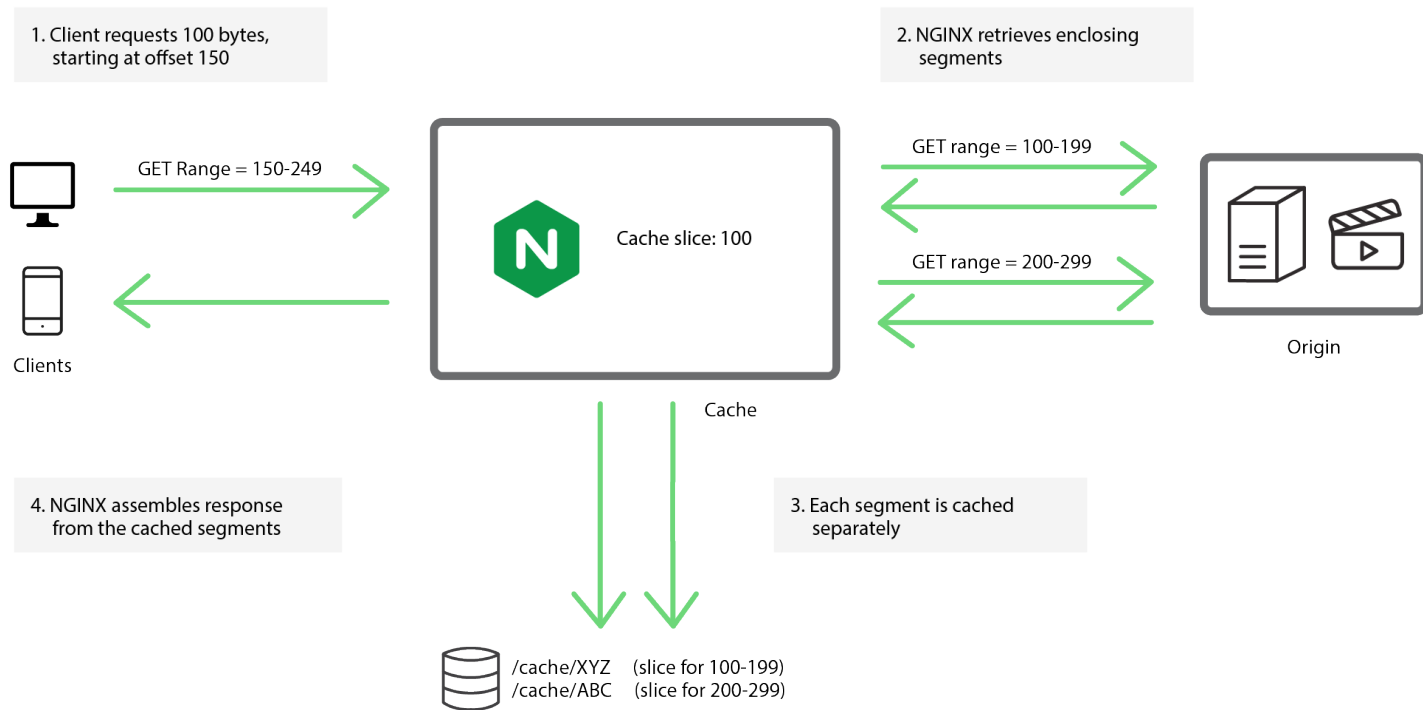
Syntax: `proxy_cache_background_update on | off;`

Default: `proxy_cache_background_update off;`

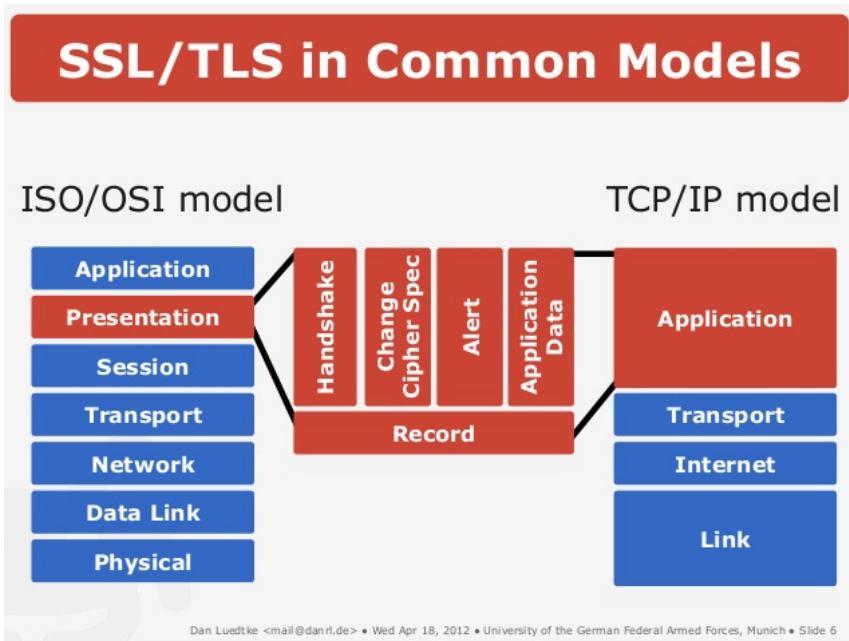
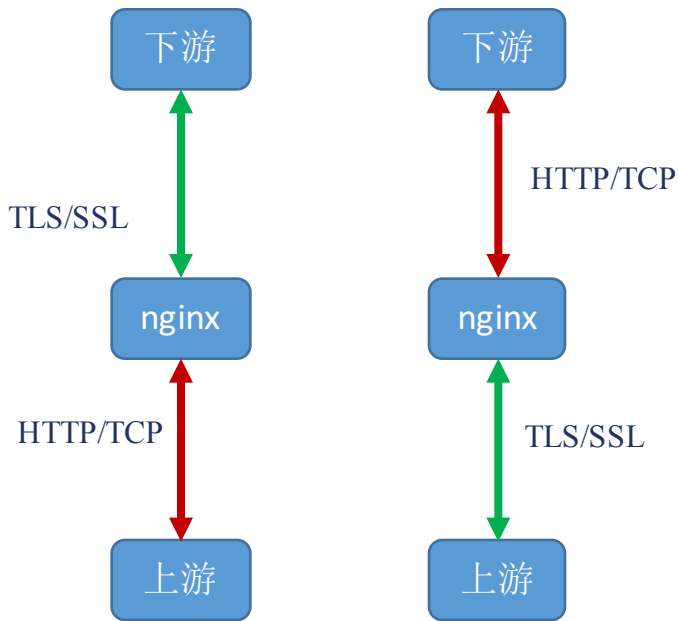
Context: `http, server, location`



slice模块：提升缓存效率



高效卸载/重装TLS层



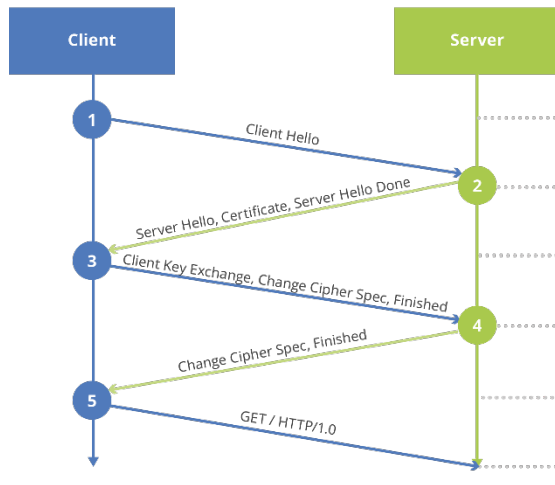
Nginx处理TLS层的高效率

- 代码的高效：C语言及设计风格

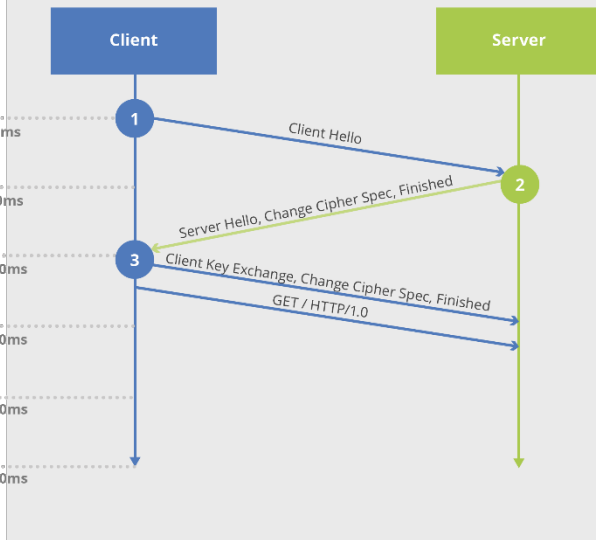
- 减少握手次数

- session缓存
- 支持ticket票据

Without resume (full handshake)



With resume (abbreviated handshake)



目录

1 大规模分布式集群的特点

2 Nginx与scalability

3 Nginx与performance

➔ 4 巧用Nginx

Nginx的四类HTTP模块

- 请求处理模块
 - 生成响应或者影响后续处理模块
- 过滤模块
 - 改变处理模块生成的响应
- 仅影响变量的模块：为其他模块赋能
 - 提供新变量，或者修改已有变量
- 负载均衡模块
 - 选择上游的负载均衡算法
 - 管理上游连接

uniform pipe and filter architecture

- 简单性
- 可重用性，可任意组合
- 可扩展性
- 可进化性
- 可验证性
- 可并发

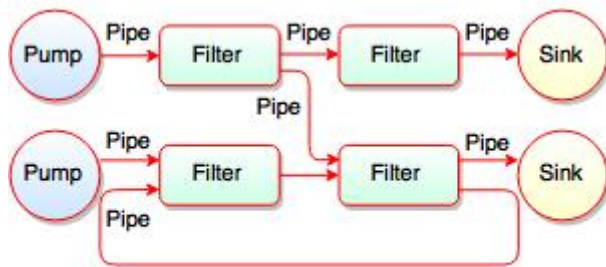
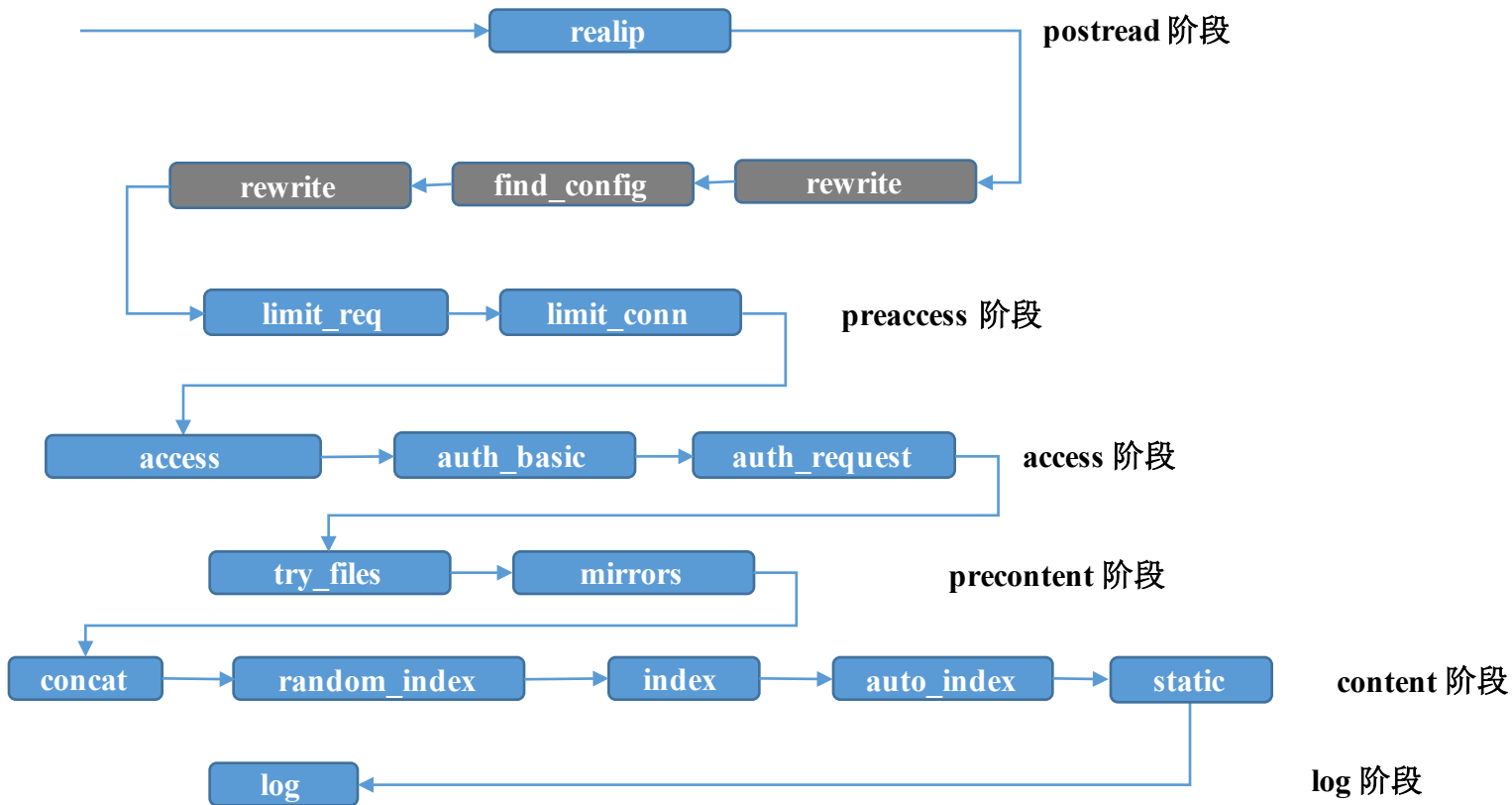



Fig. Pipes and Filters

过滤器管道架构下的HTTP处理模块



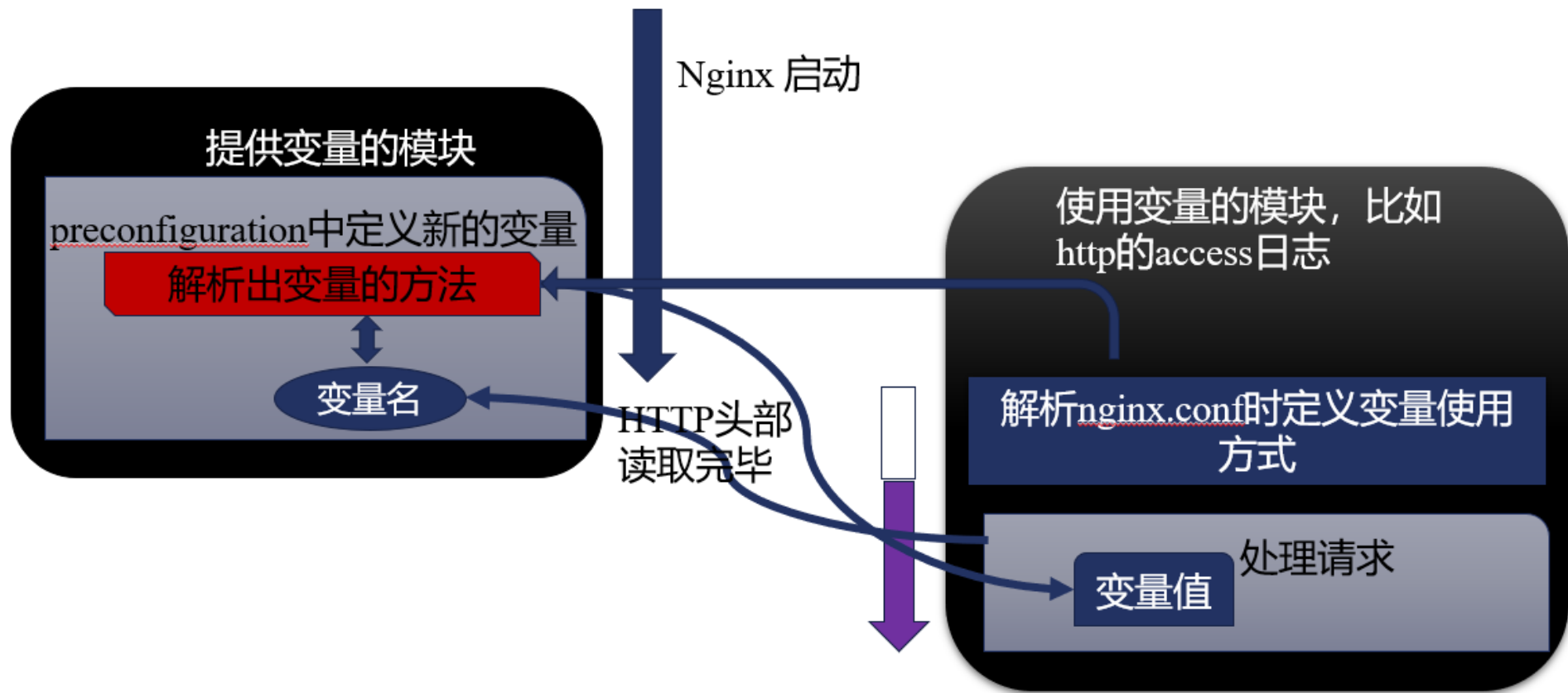
过滤器管道架构下的HTTP过滤模块

- 仅处理HTTP Response
- 通用接口
- 模块间有序
- 非关键模块可插拔

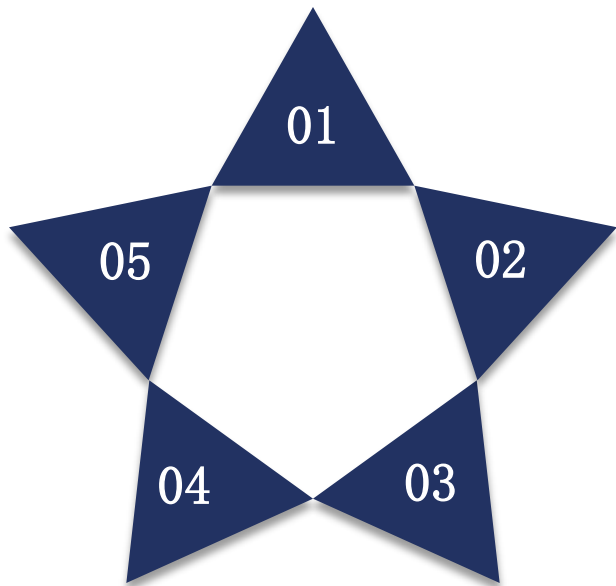


```
&ngx_http_write_filter_module,  
&ngx_http_header_filter_module,  
&ngx_http_chunked_filter_module,  
&ngx_http_v2_filter_module,  
&ngx_http_range_header_filter_module,  
&ngx_http_gzip_filter_module,  
&ngx_http_postpone_filter_module,  
&ngx_http_ssi_filter_module,  
&ngx_http_charset_filter_module,  
&ngx_http_sub_filter_module,  
&ngx_http_addition_filter_module,  
&ngx_http_userid_filter_module,  
&ngx_http_headers_filter_module,  
&ngx_http_echo_module,  
&ngx_http_xss_filter_module,  
&ngx_http_srcache_filter_module,  
&ngx_http_lua_module,  
&ngx_http_headers_more_filter_module,  
&ngx_http_rds_json_filter_module,  
&ngx_http_rds_csv_filter_module,  
&ngx_http_copy_filter_module,  
&ngx_http_range_body_filter_module,  
&ngx_http_not_modified_filter_module,
```

变量的原理



Nginx框架中的变量



01 HTTP 请求相关的变量

02 TCP 连接相关的变量

03 Nginx 处理请求过程中产生的变量

04 发送 HTTP 响应时相关的变量

05 Nginx 系统变量

Openresty的组成

Openresty

Openresty工具

lua语言模块

Nginx

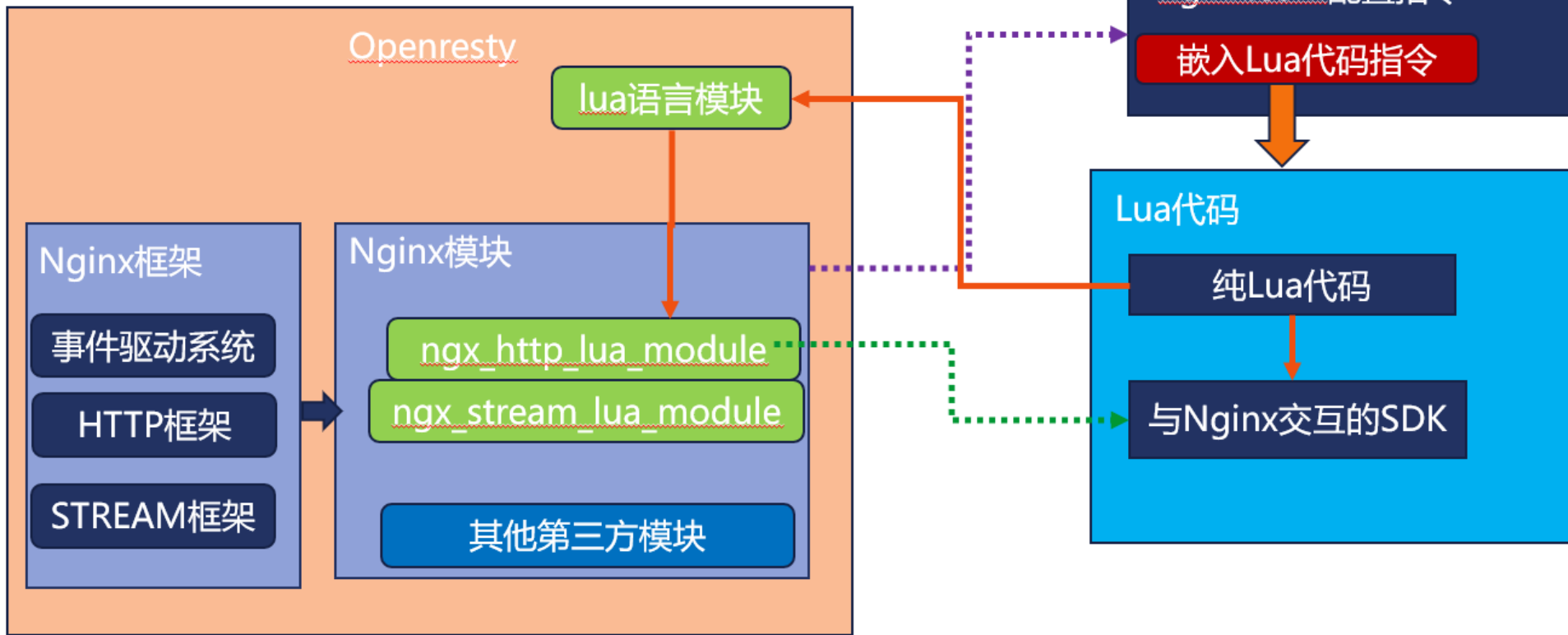
Nginx官方模块

Openresty第三方模块

ngx_http_lua_module

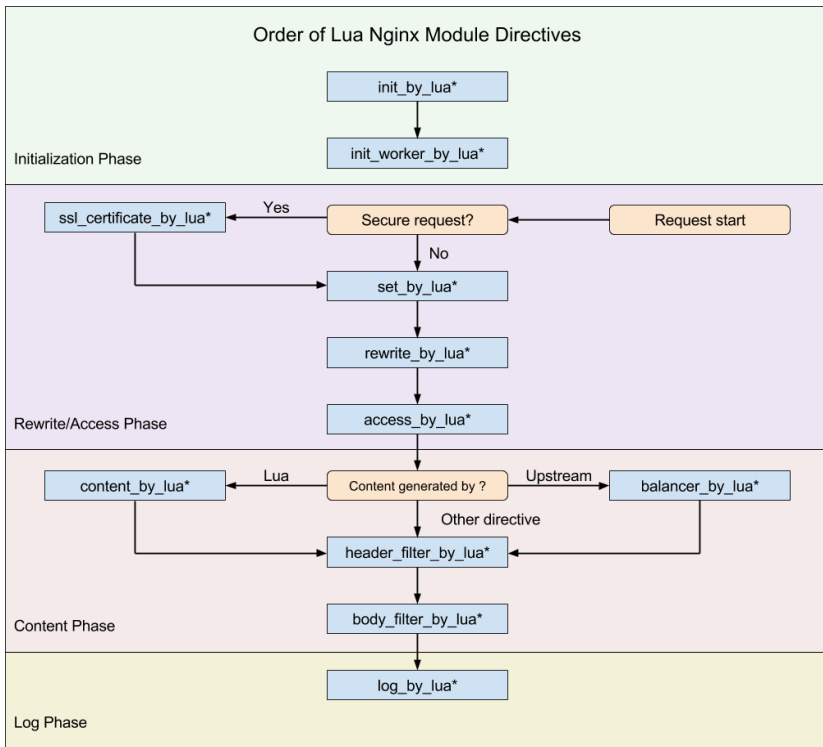
ngx_stream_lua_module

Lua代码的支持



Openresty中的SDK与指令

- cosocket通讯
 - udp
 - tcp
- 基于共享内存的字典shared.DICT
- 定时器
- 基于协程的并发编程
- 获取客户端请求与响应的信息
- 修改客户端请求与响应，包括发送响应
- 子请求
- 工具类
 - 正则表达式
 - 日志
 - 系统配置
 - 编解码
 - 时间





Thanks

高效运维社区
开放运维联盟

荣誉出品

想第一时间看到高效运维社区的
最新动态吗？

