第二部分

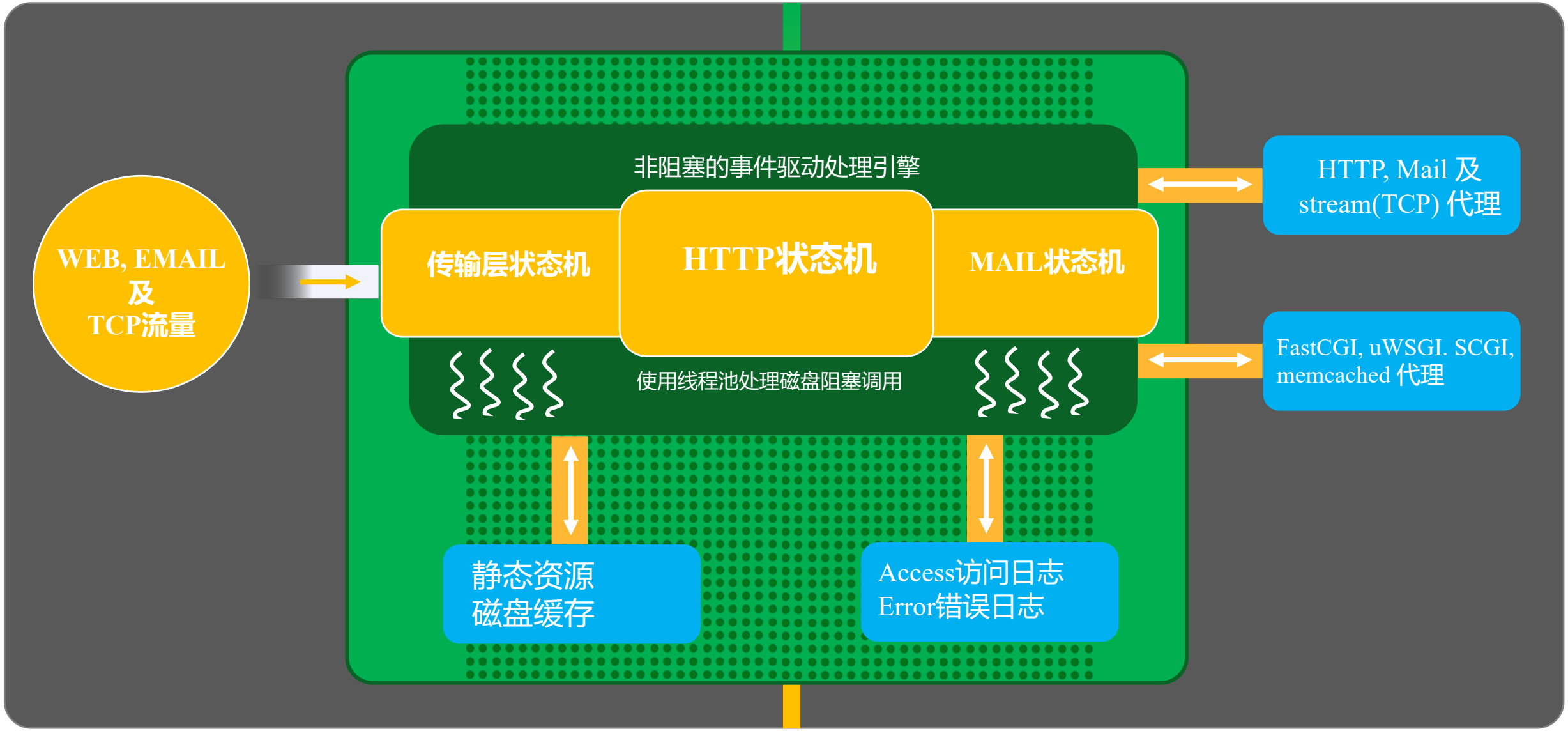
Nginx 架构基础

极客时间

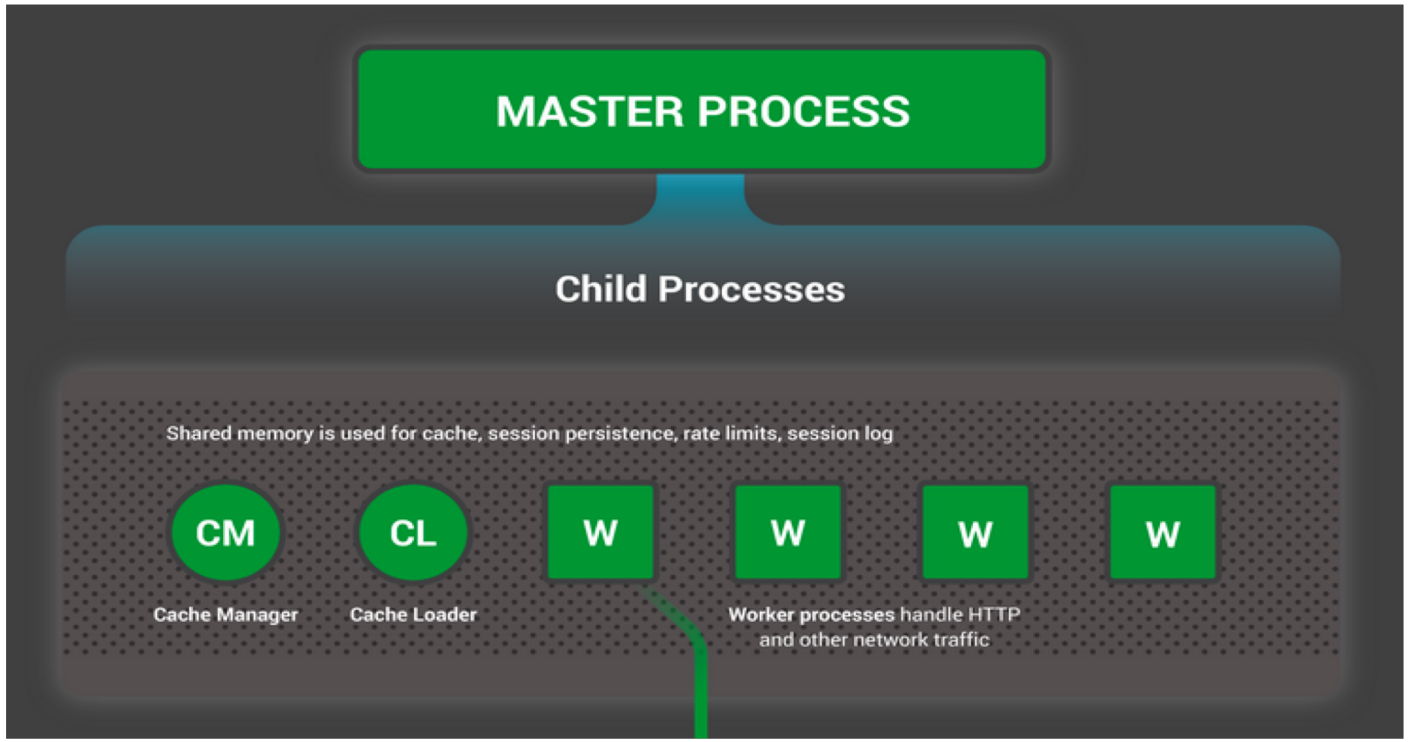
扫码试看/订阅

《Nginx 核心知识100讲》

Nginx请求处理流程



Nginx进程结构



Master进程

Worker进程

Cache manager

Cache loader

Nginx进程管理：信号

极客时间

Master进程

- 监控worker进程
  - CHLD
- 管理worker进程
- 接收信号
  - TERM, INT
  - QUIT
  - HUP
  - USR1
  - **USR2**
  - **WINCH**

Worker进程

- 接收信号
  - TERM, INT
  - QUIT
  - USR1
  - WINCH

nginx命令行

- reload：HUP
- reopen：USR1
- stop：TERM
- quit：QUIT

reload流程

极客时间

01 向master进程发送HUP信号（reload命令）

02 master进程校验配置语法是否正确

03 master进程打开新的监听端口

04 master进程用新配置启动新的worker子进程

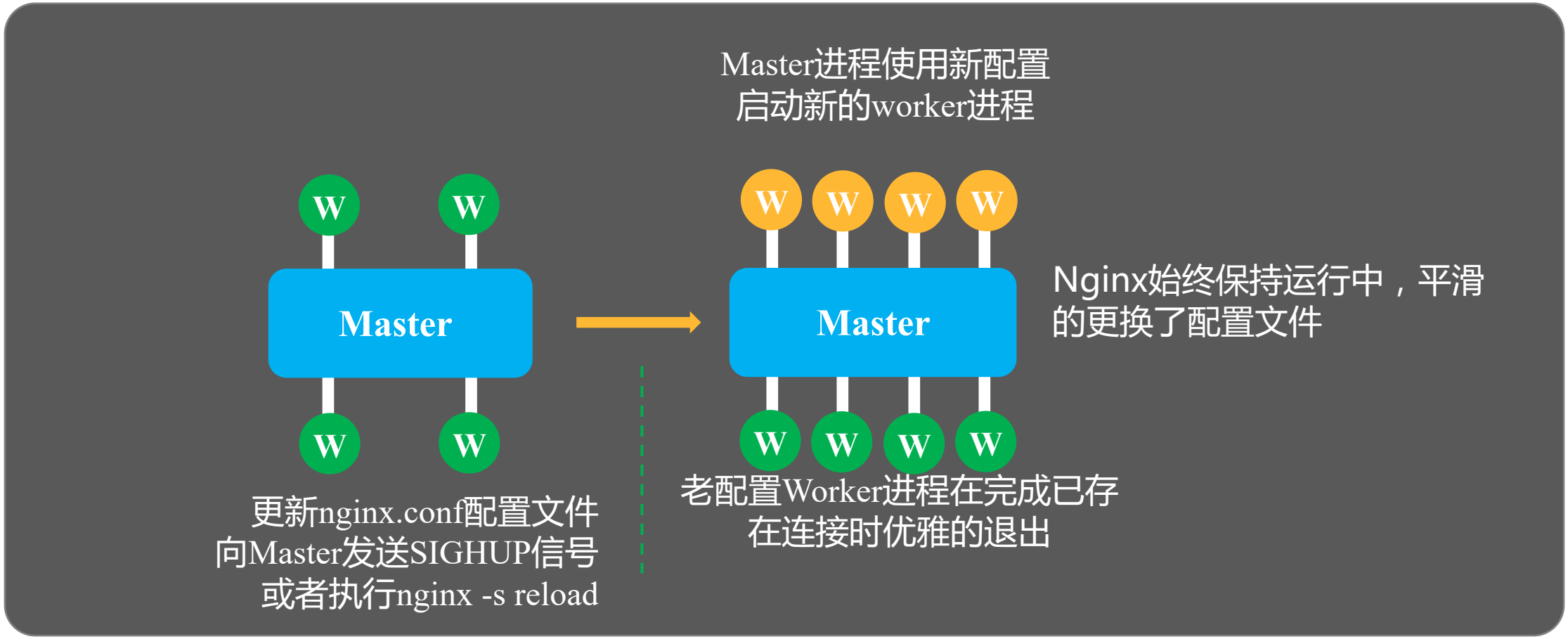
05 master进程向老worker子进程发送QUIT信号

06 老worker进程关闭监听句柄，处理完当前连接后结束进程

reload流程

极客时间

不停机载入新配置



热升级流程

极客时间

01 将旧Nginx文件换成新Nginx文件（注意备份）

02 向master进程发送USR2信号

03 master进程修改pid文件名，加后缀.oldbin

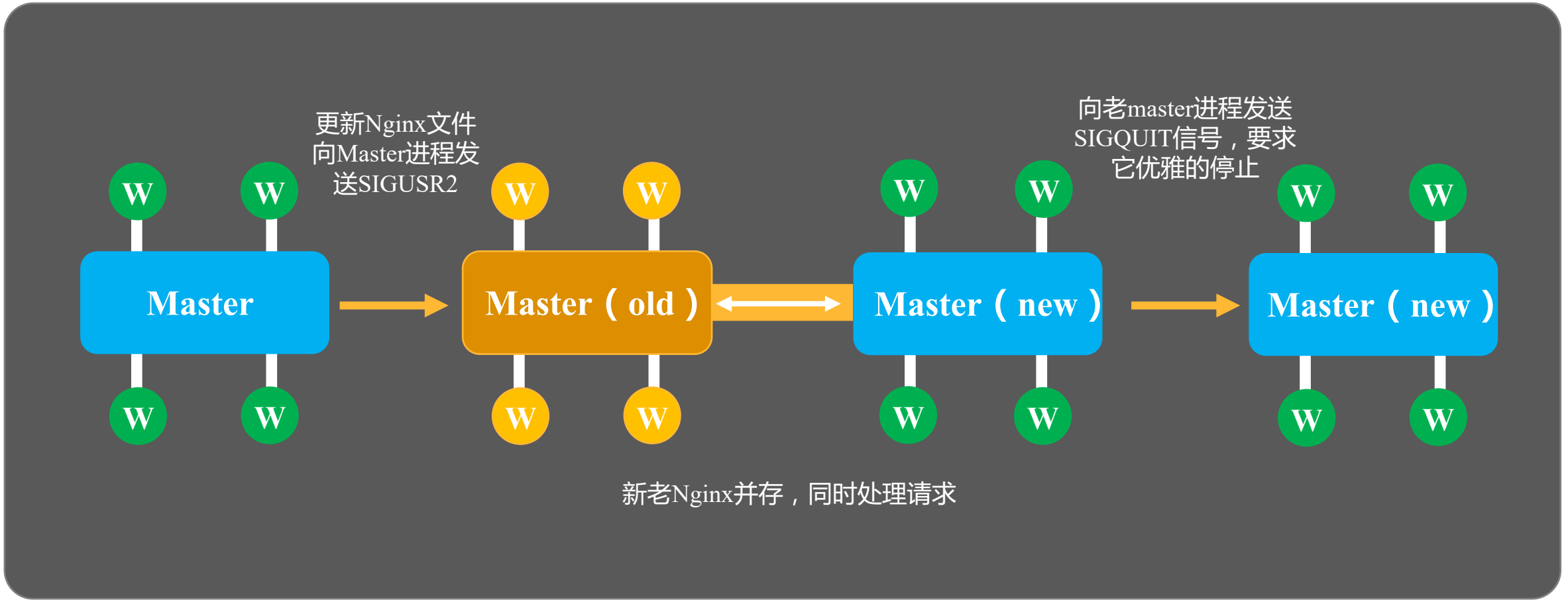
04 master进程用新Nginx文件启动新master进程

05 向老master进程发送QUIT信号，关闭老master

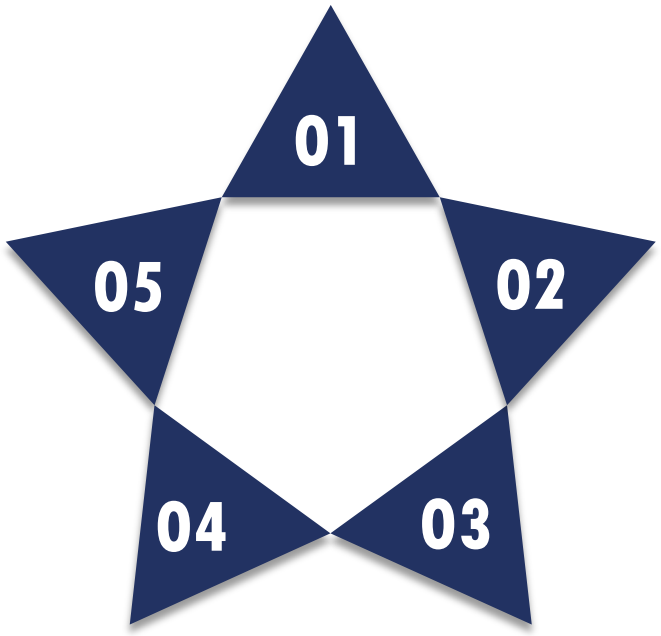
06 回滚：向老master发送HUP，向新master发送QUIT

热升级流程

不停机更新Nginx二进制文件



worker进程：优雅的关闭



01 设置定时器
worker_shutdown_timeout

02 关闭监听句柄

03 关闭空闲连接

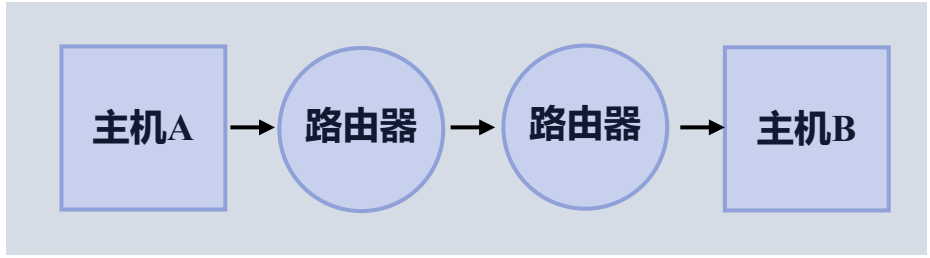
04 在循环中等待全部连接关闭

05 退出进程

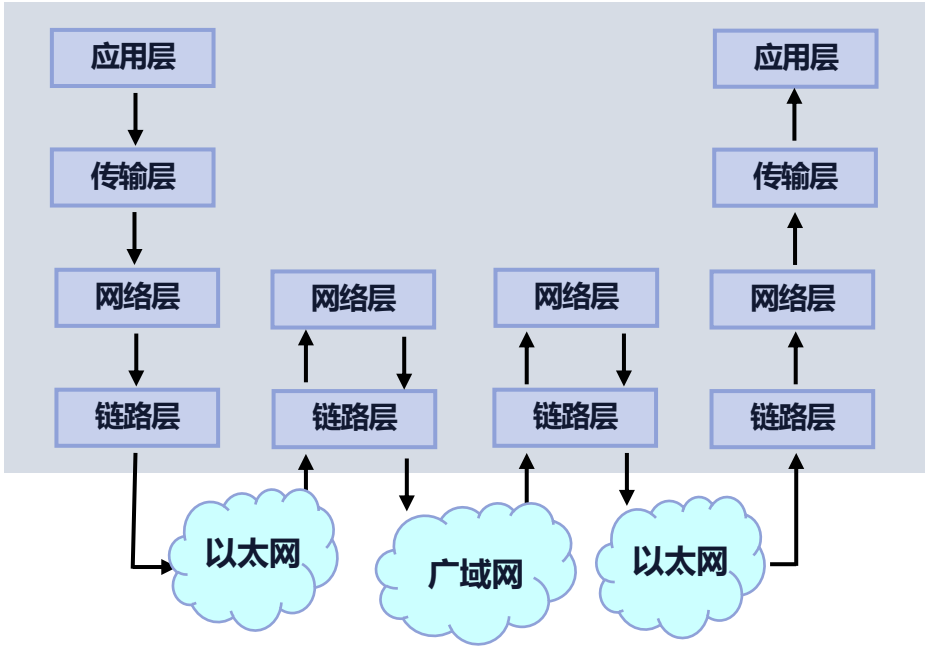
网络传输

1. 应用层
2. 传输层
3. 网络层
4. 链路层
5. 物理层

网络拓扑

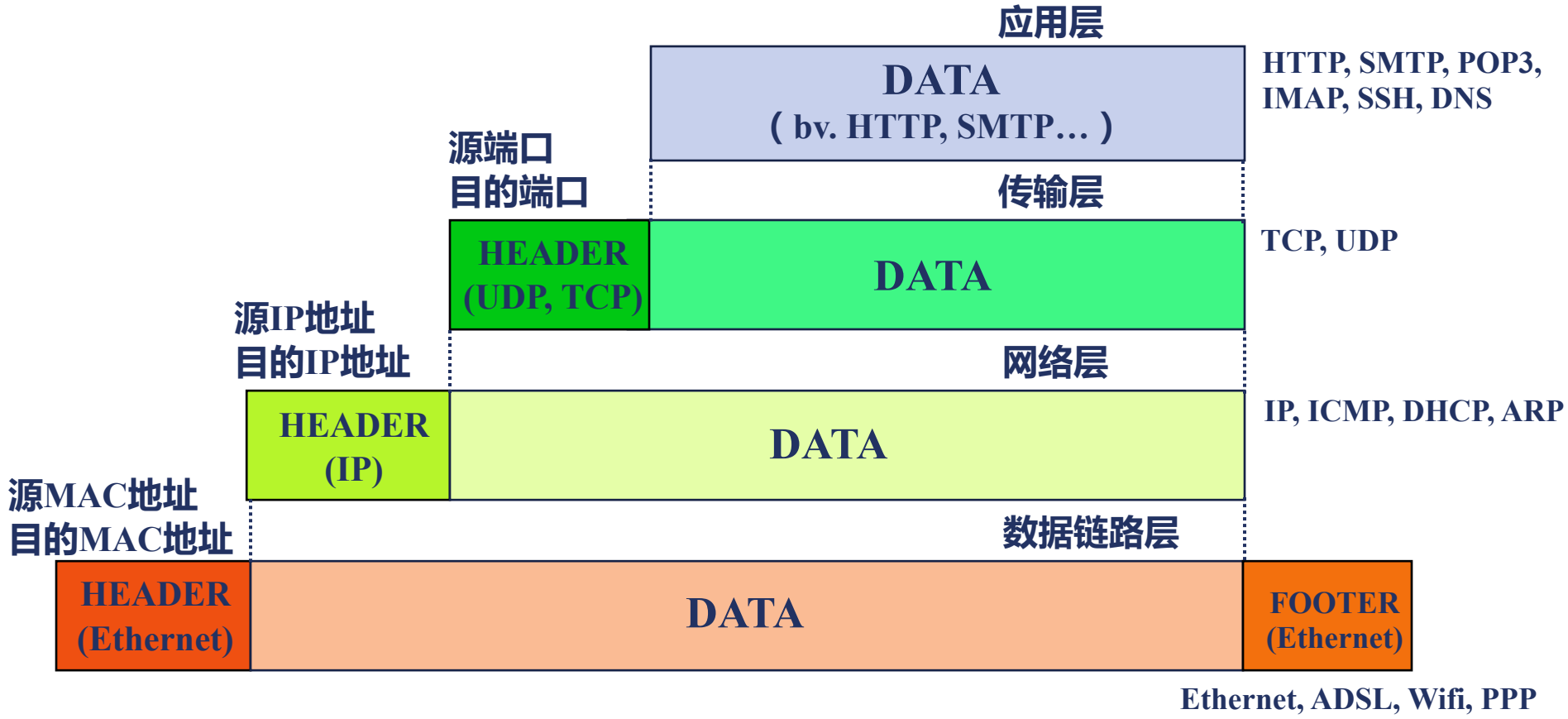


数据流

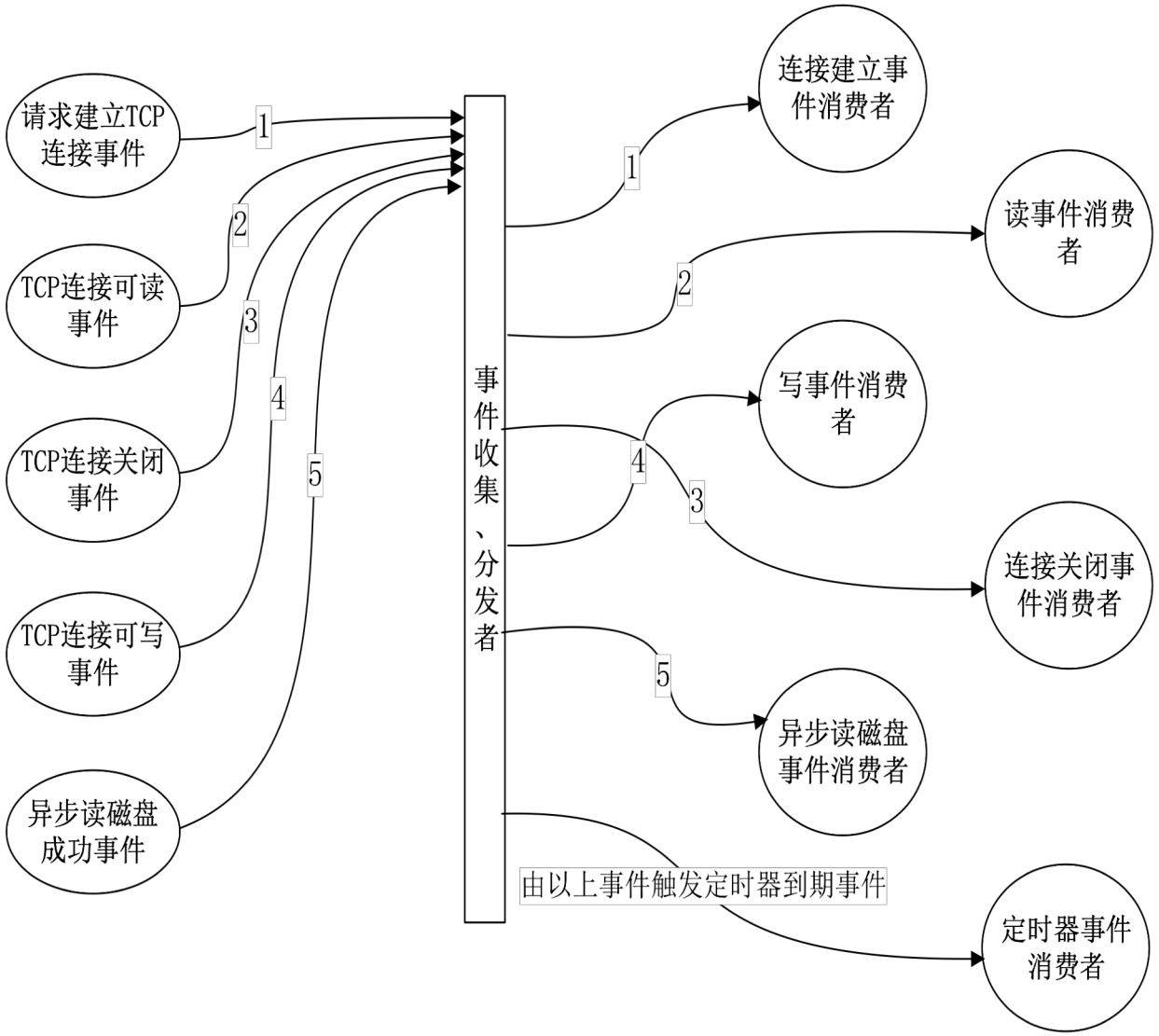


TCP流与报文

TCP/IP协议层级



TCP协议与非阻塞接口



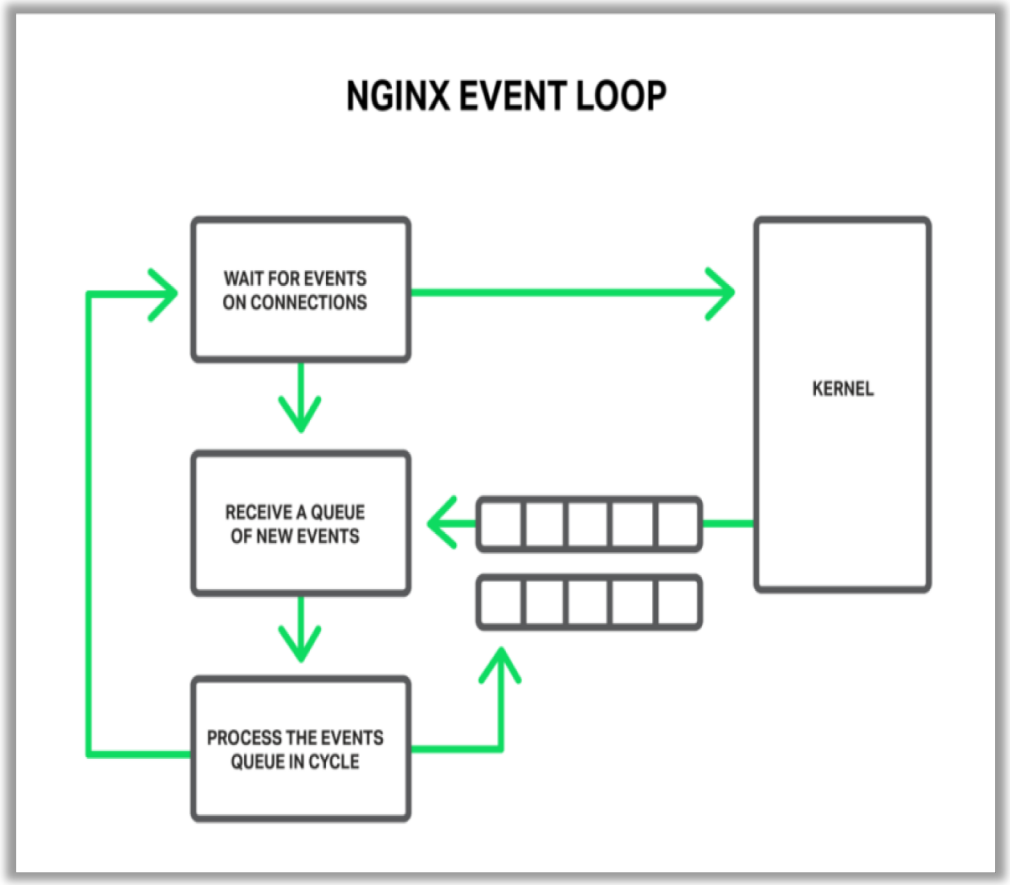
读事件

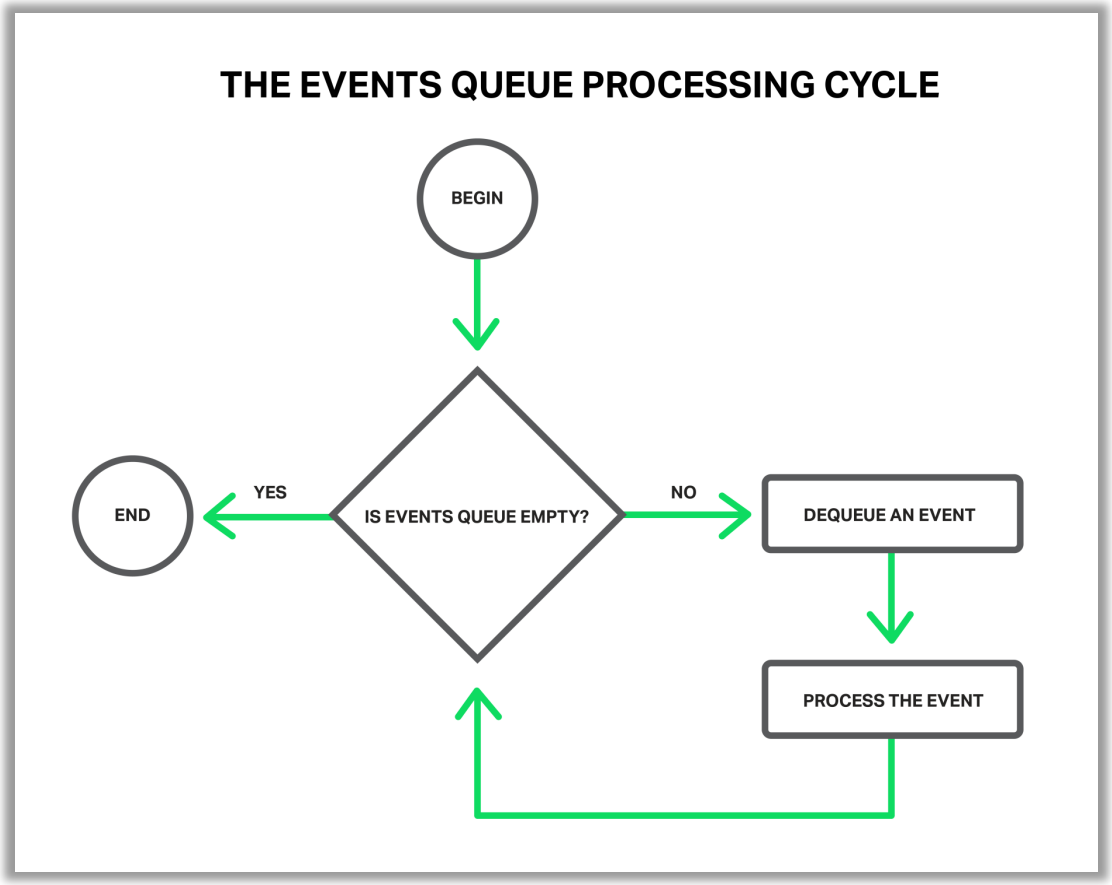
- Accept建立连接
- Read读消息

写事件

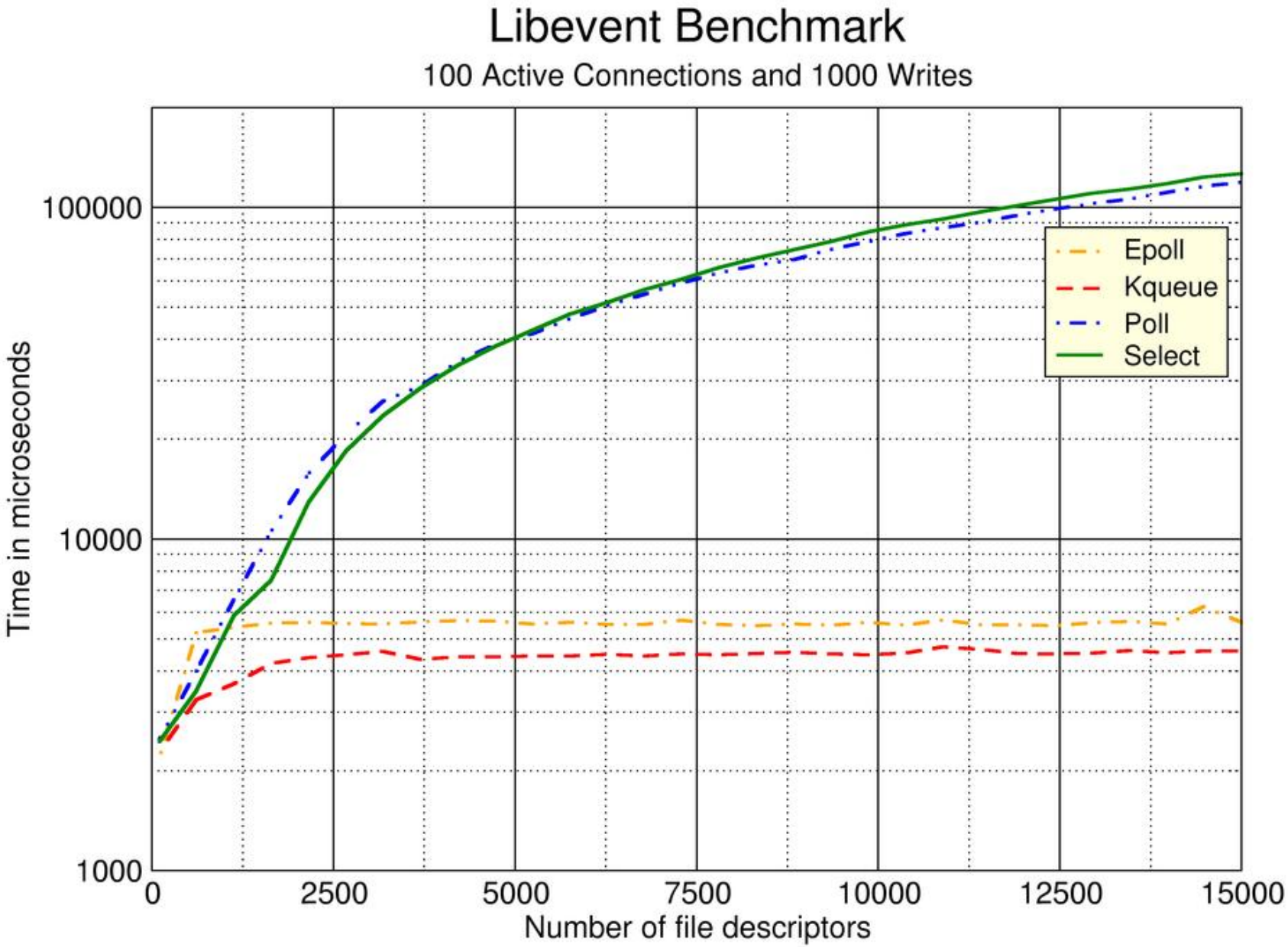
- Write写消息

Nginx事件循环


NGINX EVENT LOOP
WAIT FOR EVENTS ON CONNECTIONS
KERNEL
RECEIVE A QUEUE OF NEW EVENTS
PROCESS THE EVENTS QUEUE IN CYCLE


THE EVENTS QUEUE PROCESSING CYCLE
BEGIN
END
YES
IS EVENTS QUEUE EMPTY?
NO
DEQUEUE AN EVENT
PROCESS THE EVENT

极客时间


Libevent Benchmark
100 Active Connections and 1000 Writes
Time in microseconds
100000
10000
1000
0
2500
5000
7500
10000
12500
15000
Number of file descriptors
Epoll
Kqueue
Poll
Select

epoll

前提

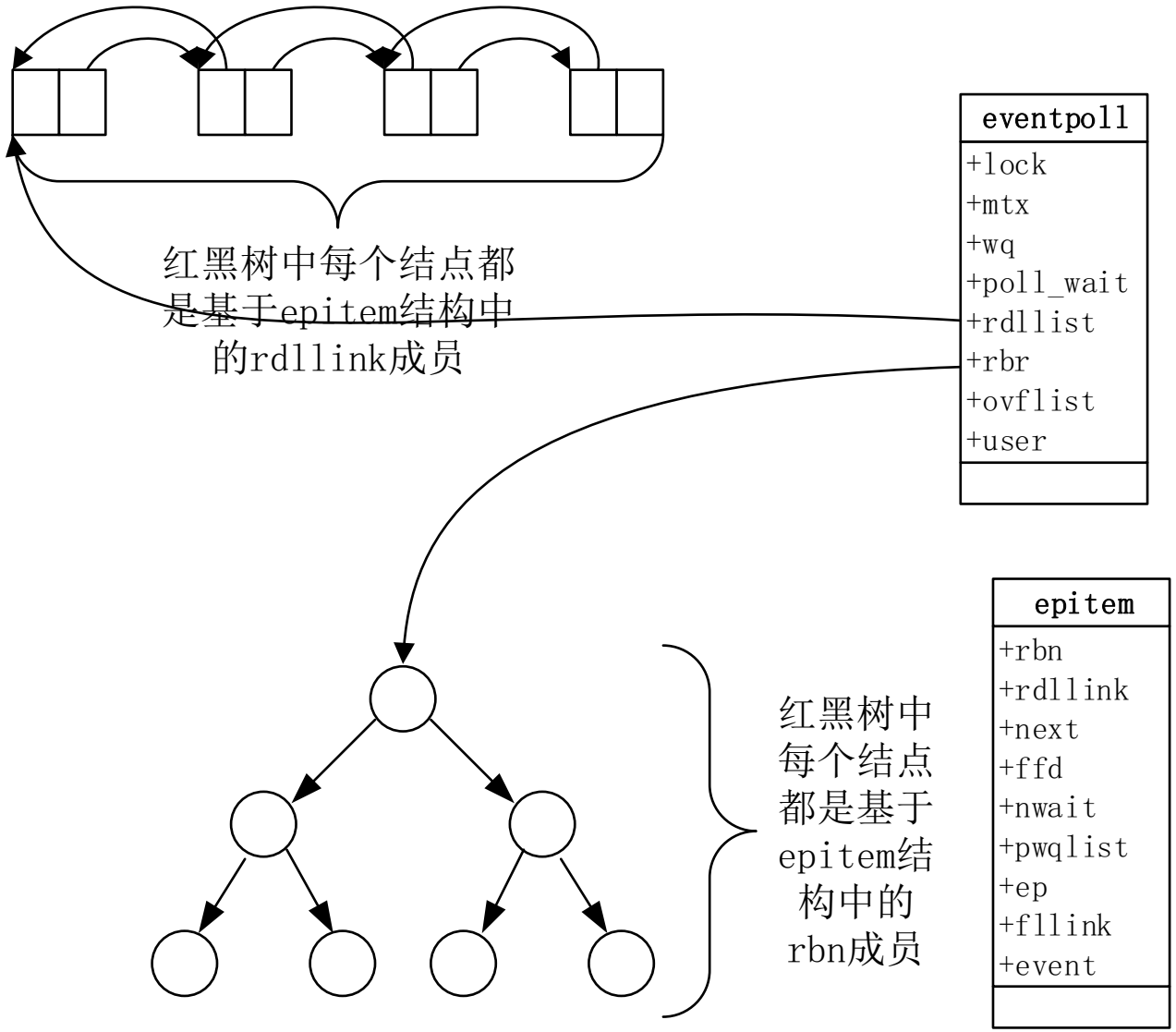
- 高并发连接中，每次处理的活跃连接数量占比很小

实现

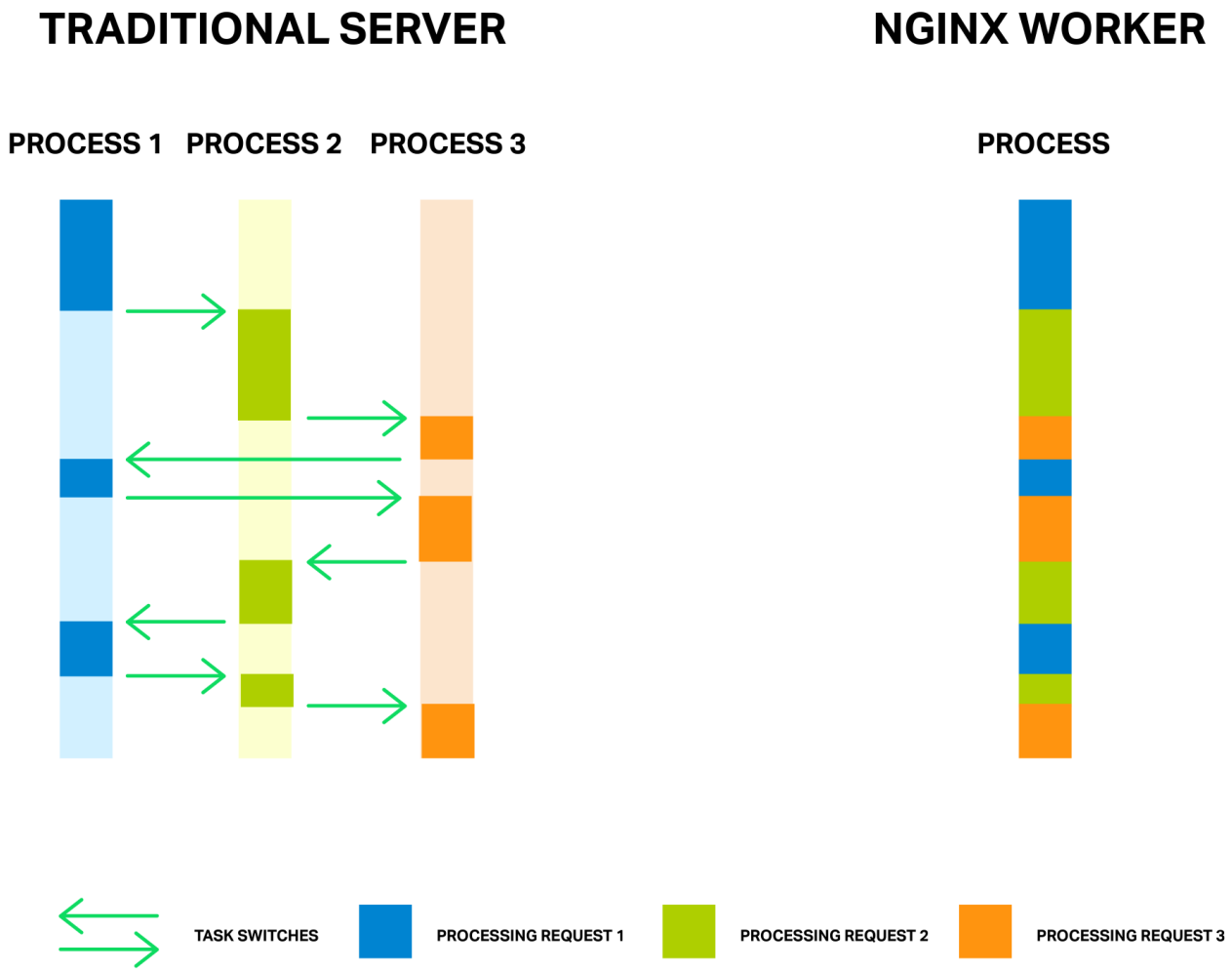
- 红黑树
- 链表

使用

- 创建
- 操作：添加/修改/删除
- 获取句柄
- 关闭



请求切换



一线程仅处理一连接

- 不做连接切换
- 依赖OS的进程调度实现并发

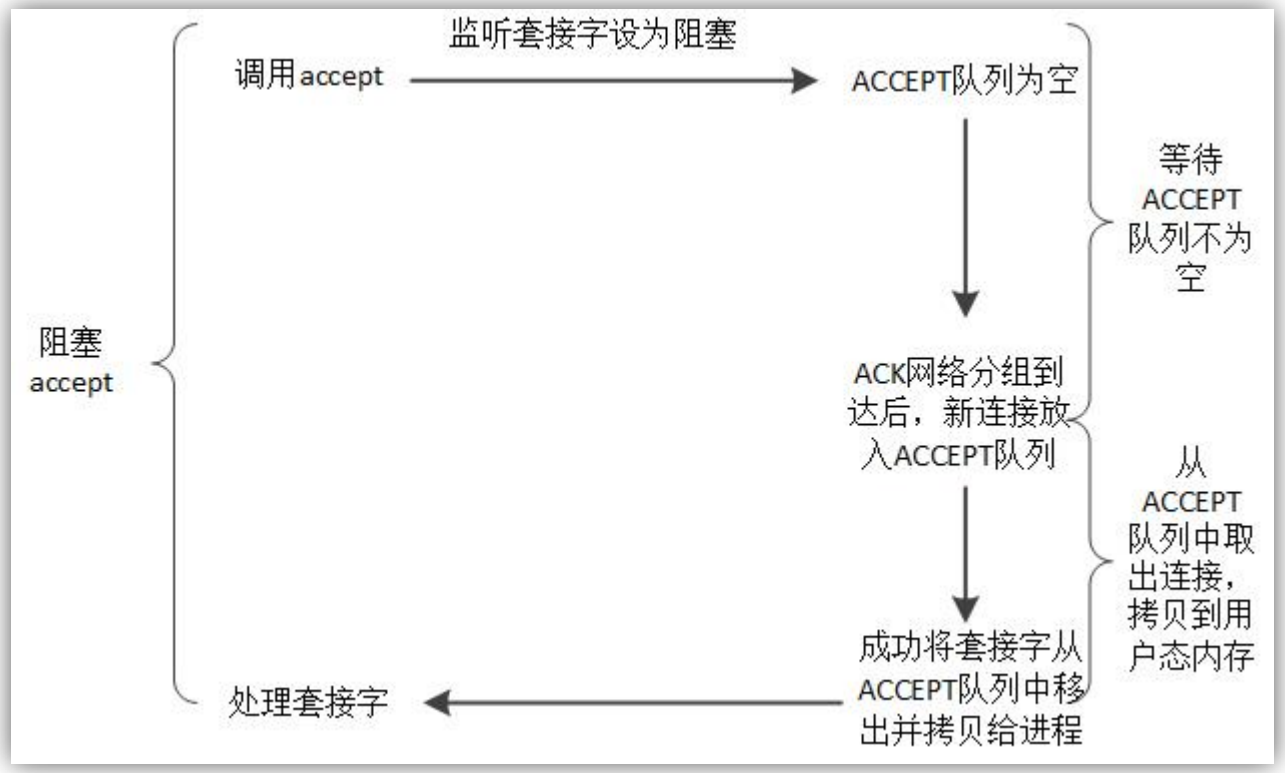
一线程同时处理多连接

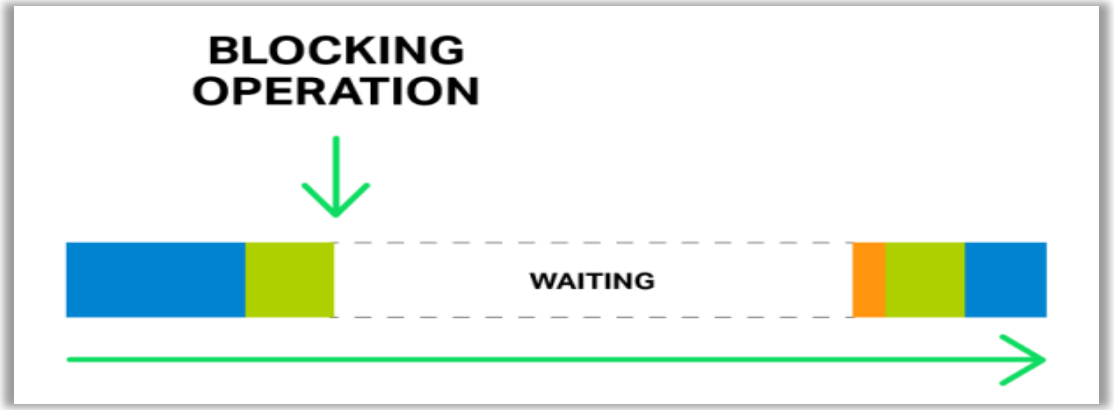
- 用户态代码完成连接切换
- 尽量减少OS进程切换

阻塞调用

极客时间

以accept为例



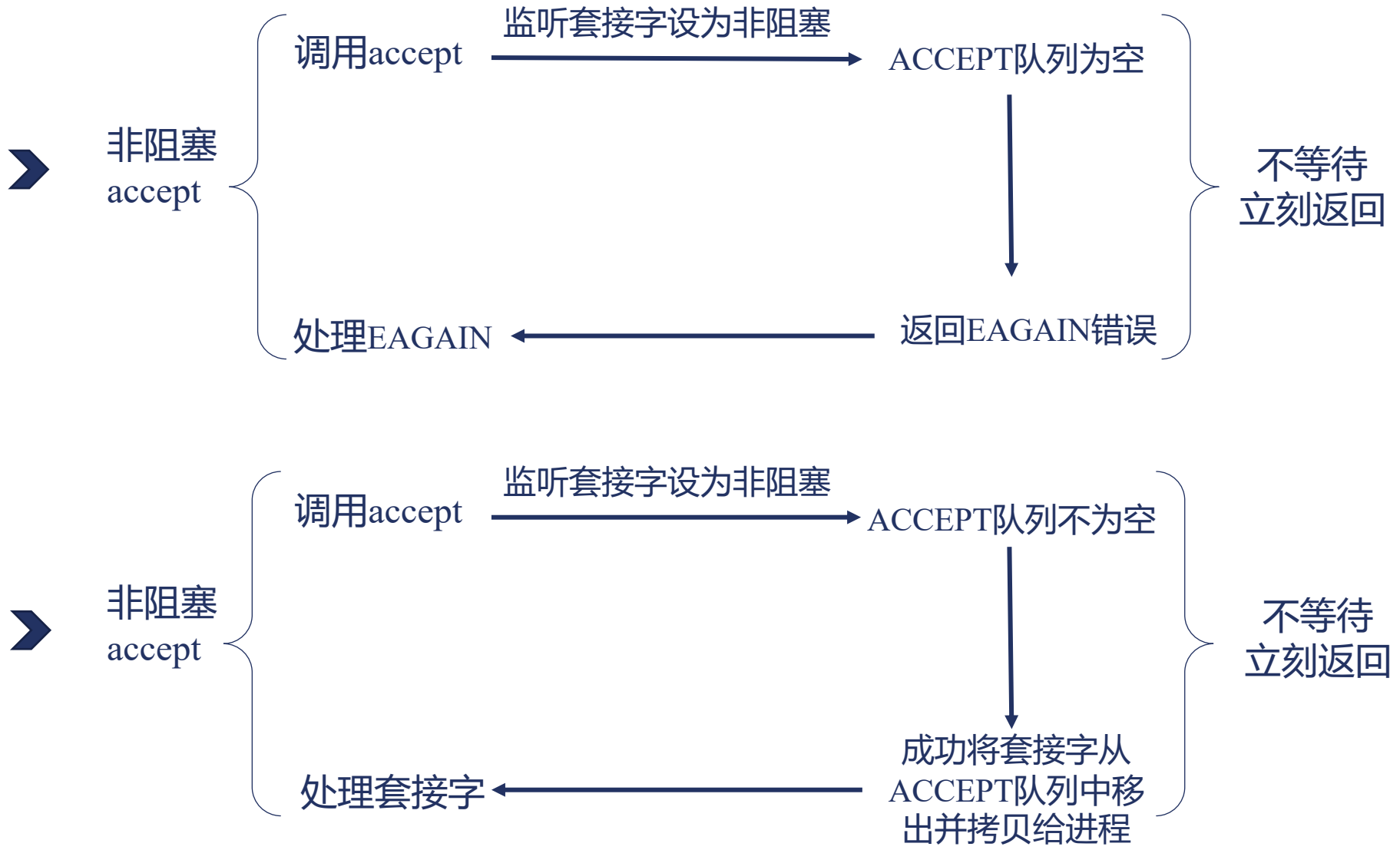


产生进程主动切换

非阻塞调用

极客时间

由你的代码决定是否切换新任务



非阻塞调用下的同步与异步

极客时间

Openresty的同步调用代码

```
local client = redis:new()
 client:set_timeout(30000)
 local ok,err = client:connect(ip,port)
 if not ok then
        ngx.say("failed: ",err)
        return
 end
```

这是标准的异步调用

```
rc = ngx_http_read_request_body ( r, ngx_http_upstream_init) ;
if ( rc > = NGX_HTTP_SPECIAL_RESPONSE) {
    return rc ;
}
```

这个方法执行完时调用post_handler异步方法

```
ngx_int_t
ngx_http_read_client_request_body (ngx_http_request_t  * r,
    ngx_http_client_body_handler_pt  post_handler)
```

最终读取完body后调用ngx_http_upstream_init方法

```
void
ngx_http_upstream_init (ngx_http_request_t  * r )
{
```

Nginx 模块

前提
编译进Nginx

提供哪些
配置项

模块何时
被使用

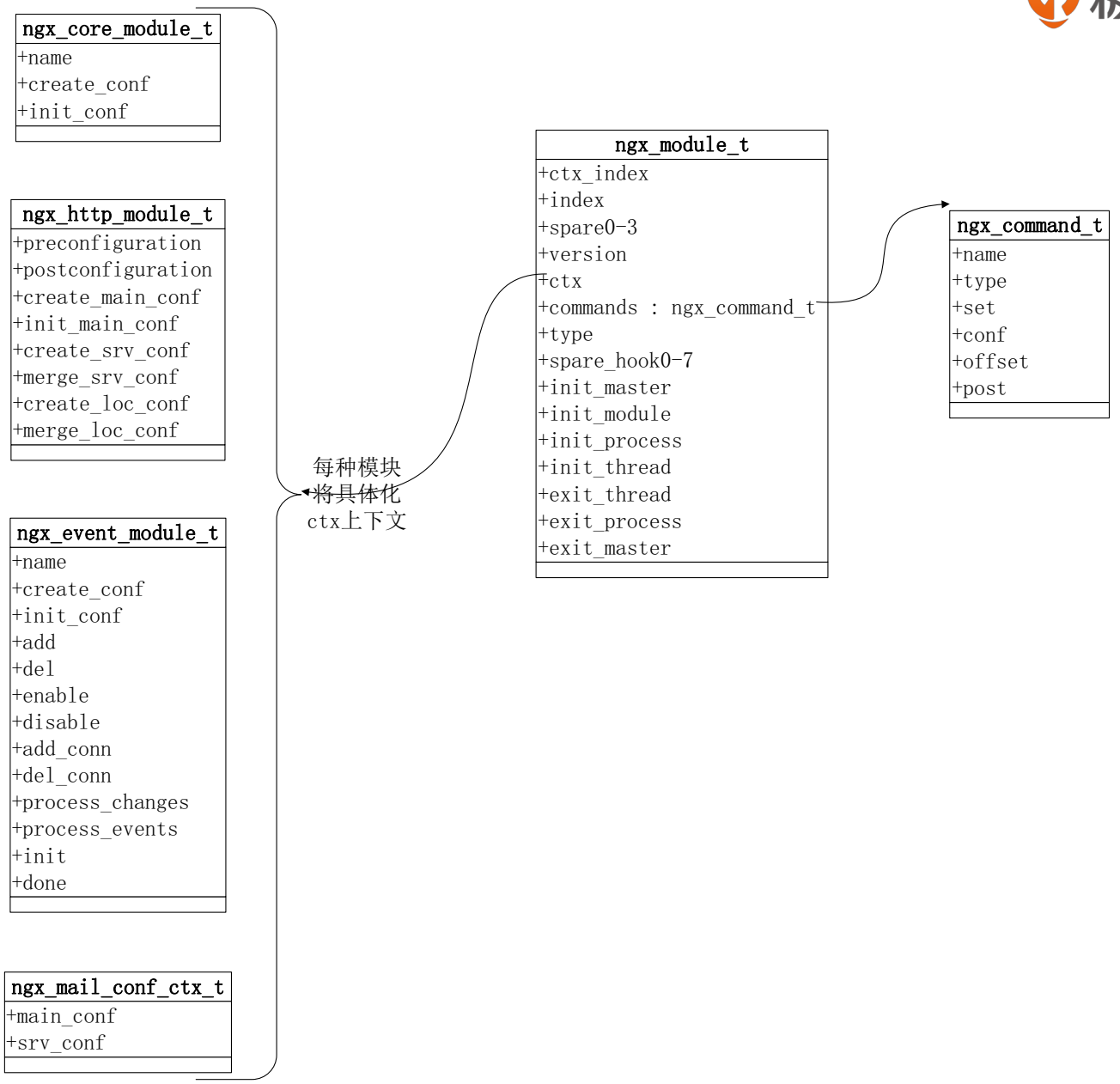
提供哪些
变量

Nginx 模块

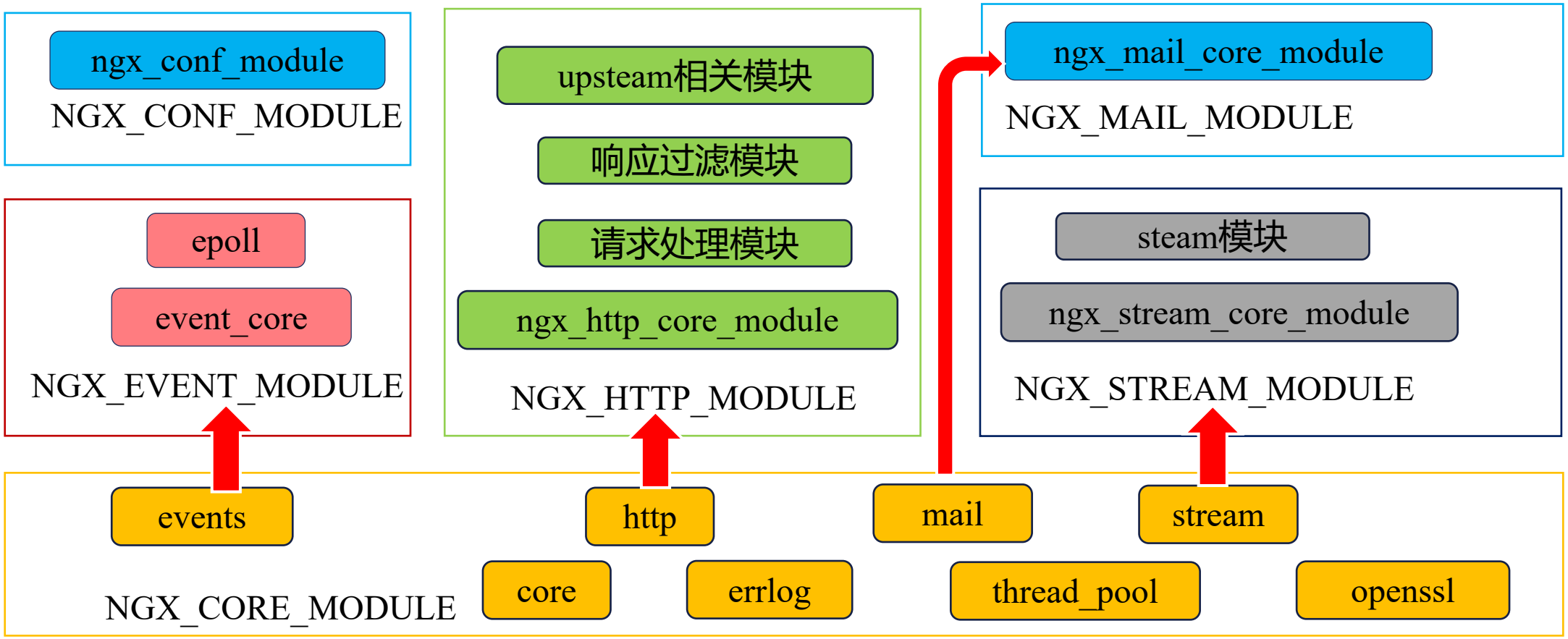
内聚

抽象

- 配置
- 启停回调方法
- 子模块抽象
  - http
  - event
  - mail
  - stream



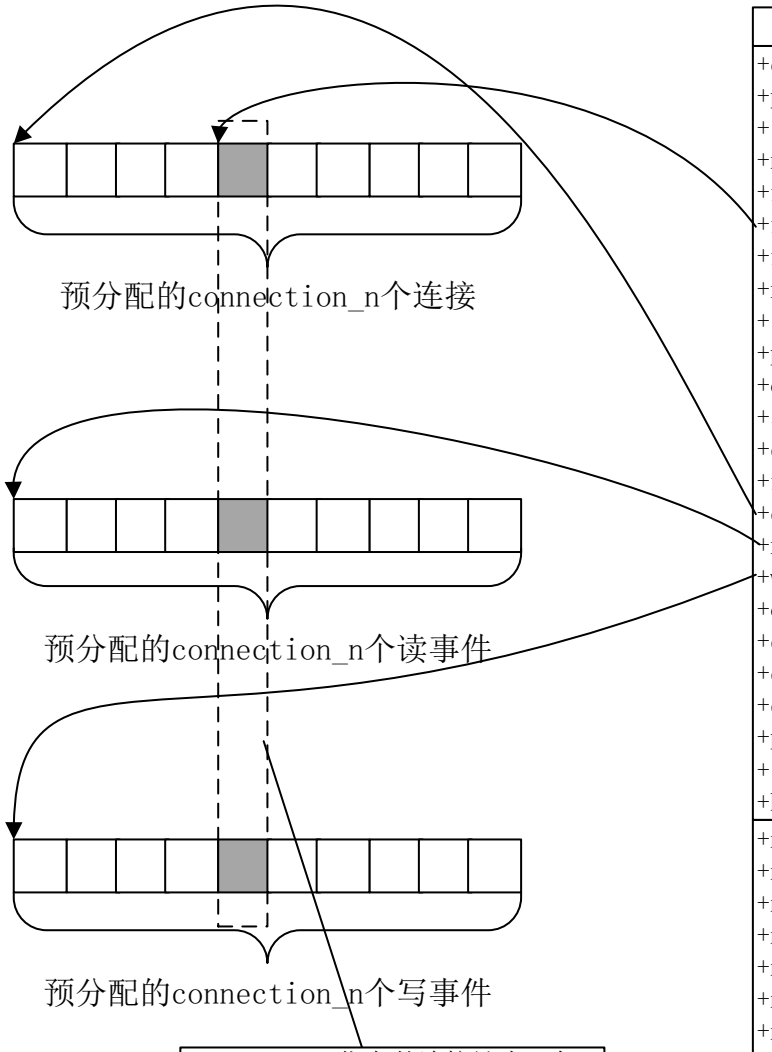
模块分类


ngx_conf_module
NGX_CONF_MODULE
epoll
event_core
NGX_EVENT_MODULE
upsteam相关模块
响应过滤模块
请求处理模块
ngx_http_core_module
NGX_HTTP_MODULE
ngx_mail_core_module
NGX_MAIL_MODULE
steam模块
ngx_stream_core_module
NGX_STREAM_MODULE
events
http
mail
stream
core
errlog
thread_pool
openssl
NGX_CORE_MODULE

连接池

构成

- 对下游客户端的连接
- 对上游服务器的连接



ngx_cycle_t
+conf_ctx +pool +log +new_log +files +free_connections +free_connection_n +reusable_connections_queue : ngx_queue_s +listening : ngx_array_t +pathes : ngx_array_t +open_files : ngx_list_t +shared_memory : ngx_list_t +connection_n +files_n +connections +read_events +write_events +old_cycle +conf_file +conf_param +conf_prefix +prefix +lock_file +hostname
+ngx_master_process_cycle() +ngx_single_process_cycle() +ngx_start_worker_processes() +ngx_start_cache_manager_processes() +ngx_pass_open_channel() +ngx_signal_worker_processes() +ngx_reap_children() +ngx_master_process_exit() +ngx_worker_process_cycle() +ngx_worker_process_init() +ngx_worker_process_exit() +ngx_cache_manager_process_cycle() +ngx_process_events_and_timers()

核心数据结构

```
struct ngx_event_s {
    void            *data;
    unsigned         instance:1;
    unsigned         timedout:1;
    unsigned         timer_set:1;
    unsigned         available:1;
    ngx_event_handler_pt  handler;
    ngx_uint_t       index;
    ngx_log_t       *log;
    ngx_rbtree_node_t   timer;
   ngx_queue_t      queue;
    … …
};
```

96 字节

字节 232

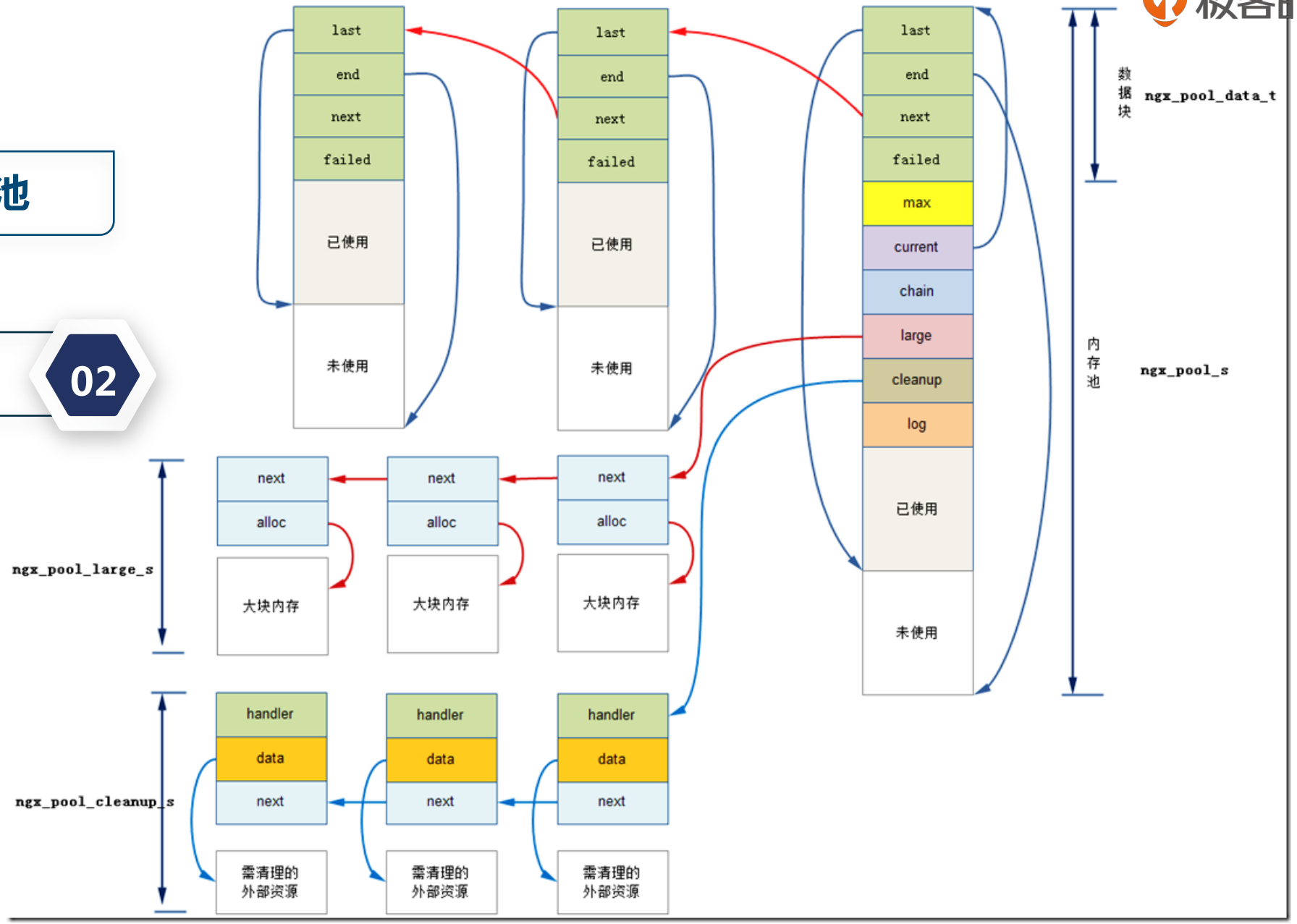
```
struct ngx_connection_s {
    void               *data;
    ngx_event_t        *read;
    ngx_event_t        *write;   读写事件
    ngx_socket_t        fd;
    ngx_recv_pt         recv;
    ngx_send_pt         send;    抽象解耦OS底层方法
    off_t               sent;          <=   bytes_sent变量
    ngx_log_t          *log;
    ngx_pool_t          *pool;  <= 初始 connection_pool_size配置
    int                 type;
    struct sockaddr     *sockaddr;
    socklen_t           socklen;
    ngx_str_t           addr_text;
    ngx_str_t           proxy_protocol_addr;
    in_port_t           proxy_protocol_port;
    ngx_buf_t           *buffer;
    ngx_queue_t         queue;
};
```

极客时间

内存池

01 连接内存池

02 请求内存池



Nginx进程间的通讯方式

极客时间

- 信号
- 共享内存

- 锁
- Slab内存管理器

共享内存：跨worker进程通讯

哪些官方nginx模块使用了共享内存

- 共享内存使用者
  - Ngx_http_lua_api
  - rbtree
    - Ngx_stream_limit_conn_module
    - Ngx_http_limit_conn_module
    - Ngx_stream_limit_req_module
    - http cache
      - Ngx_http_file_cache
      - Ngx_http_proxy_module
      - Ngx_http_scgi_module
      - Ngx_http_uwsgi_module
      - Ngx_http_fastcgi_module
    - ssl
      - Ngx_http_ssl_modul
      - Ngx_mail_ssl_module
      - Ngx_stream_ssl_module
  - 单链表
    - Ngx_http_upstream_zone_module
    - Ngx_stream_upstream_zone_module

OpenResty共享内存代码示例

极客时间

```
http {
    lua_shared_dict dogs 10m;
    server {
        location /set {
            content_by_lua_block {
                local dogs = ngx.shared.dogs
                dogs:set("Jim", 8)
                ngx.say("STORED")
            }
        }
        location /get {
            content_by_lua_block {
                local dogs = ngx.shared.dogs
                ngx.say(dogs:get("Jim"))
            }
        }
    }
}
```

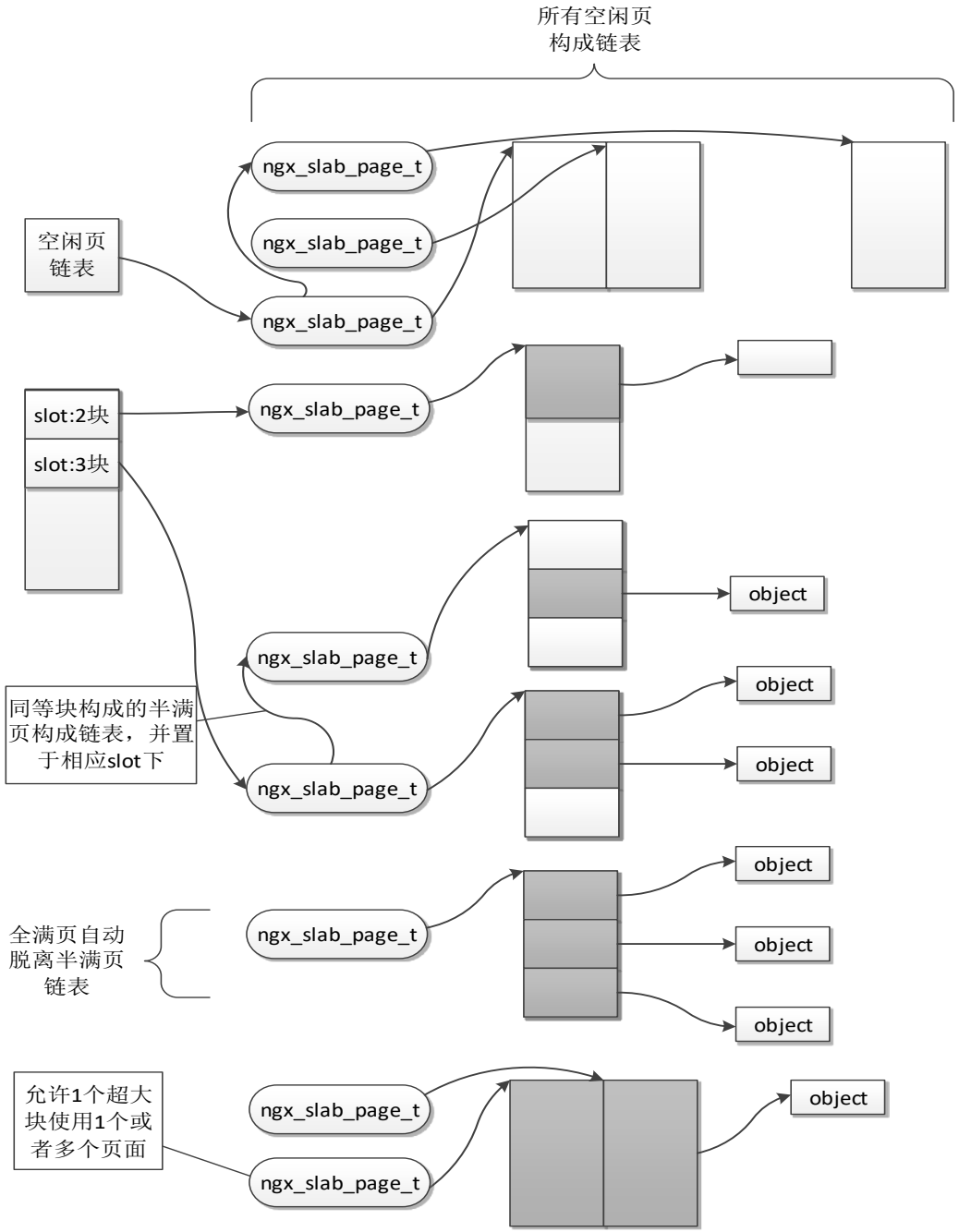
Slab内存管理

Bestfit

- 最多两倍内存消耗

Bestfit

- 适合小对象
- 避免碎片
- 避免重复初始化



ngx_slab_stat：统计Slab使用状态

极客时间

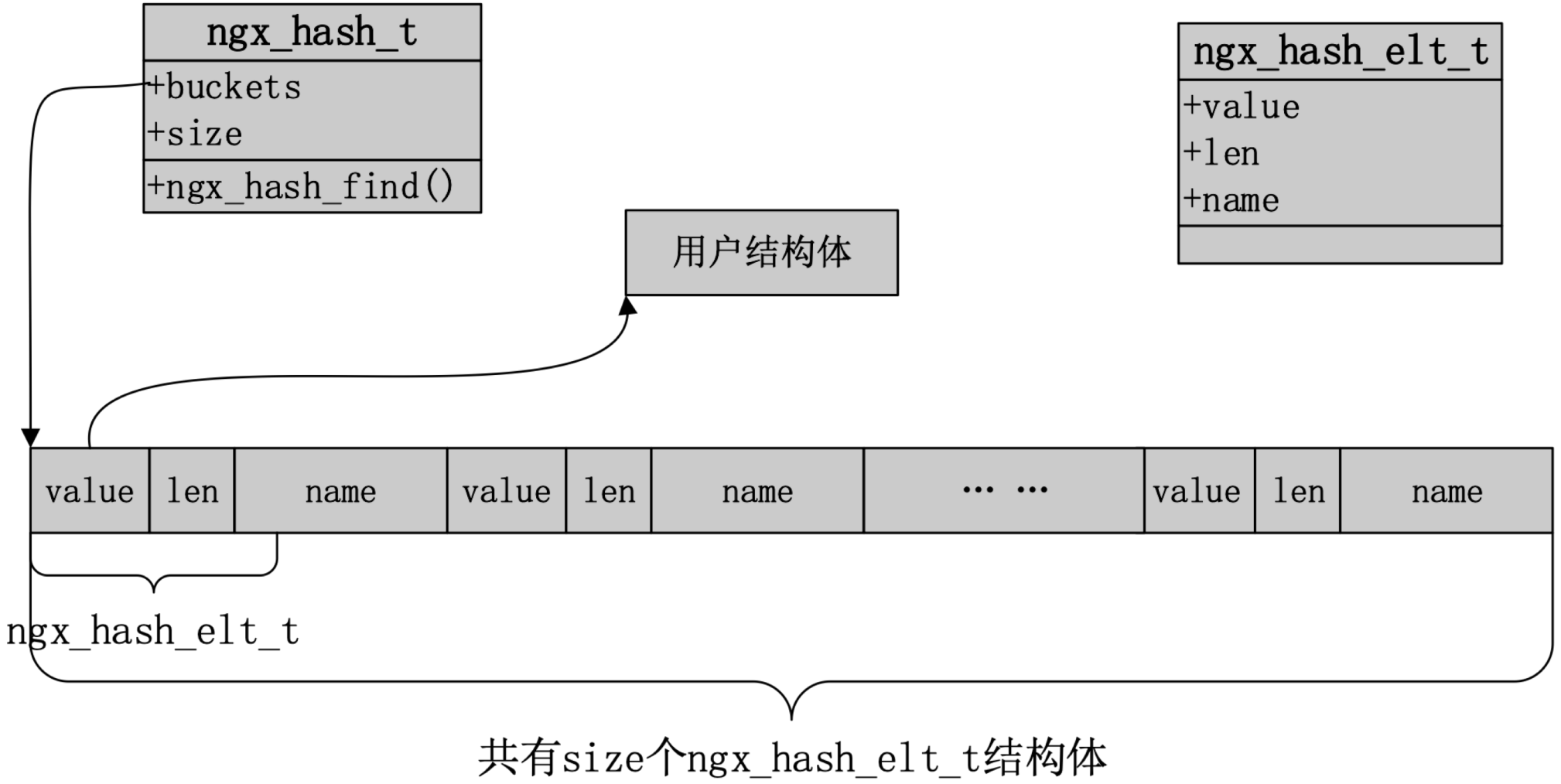
```
$ curl http://localhost:80/slab_stat
* shared memory: one
total:      102400(KB) free:      101792(KB) size:           4(KB)
pages:      101792(KB) start:0000000003496000 end:0000000009800000
slot:           8(Bytes) total:           0 used:           0 reqs:           0 fails:           0
slot:          16(Bytes) total:           0 used:           0 reqs:           0 fails:           0
slot:          32(Bytes) total:         127 used:           1 reqs:           1 fails:           0
slot:          64(Bytes) total:           0 used:           0 reqs:           0 fails:           0
slot:         128(Bytes) total:          32 used:           1 reqs:           1 fails:           0
slot:         256(Bytes) total:           0 used:           0 reqs:           0 fails:           0
slot:         512(Bytes) total:           0 used:           0 reqs:           0 fails:           0
slot:        1024(Bytes) total:           0 used:           0 reqs:           0 fails:           0
slot:        2048(Bytes) total:           0 used:           0 reqs:           0 fails:           0
```

Nginx 容器

- 数组
- 链表
- 队列
- 哈希表
- 红黑树
- 基数树

极客时间

Nginx 哈希表

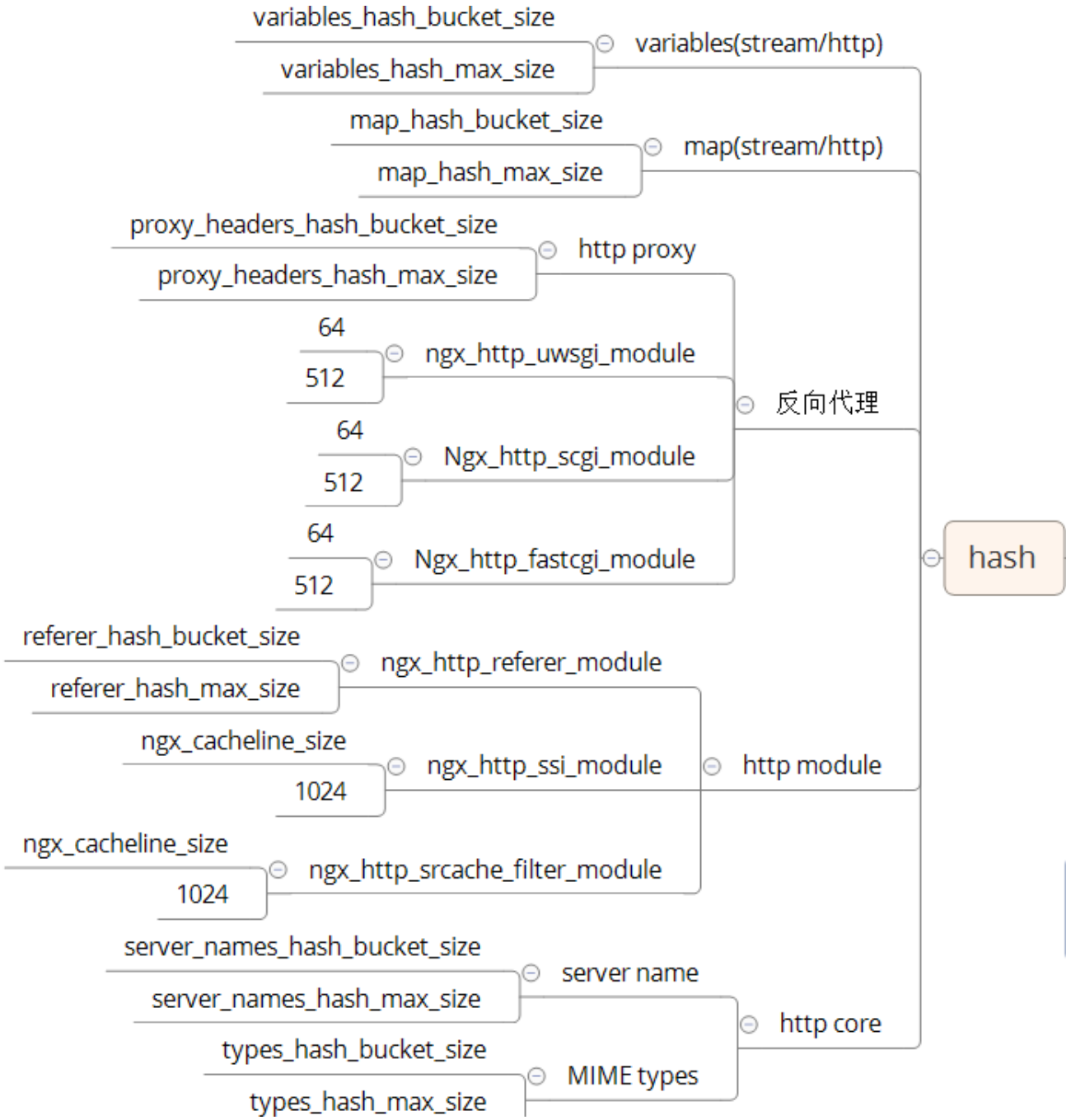

ngx_hash_t
+buckets
+size
+ngx_hash_find()
ngx_hash_elt_t
+value
+len
+name
用户结构体
value
len
name
value
len
name
… …
value
len
name
ngx_hash_elt_t
共有size个ngx_hash_elt_t结构体

哈希表配置

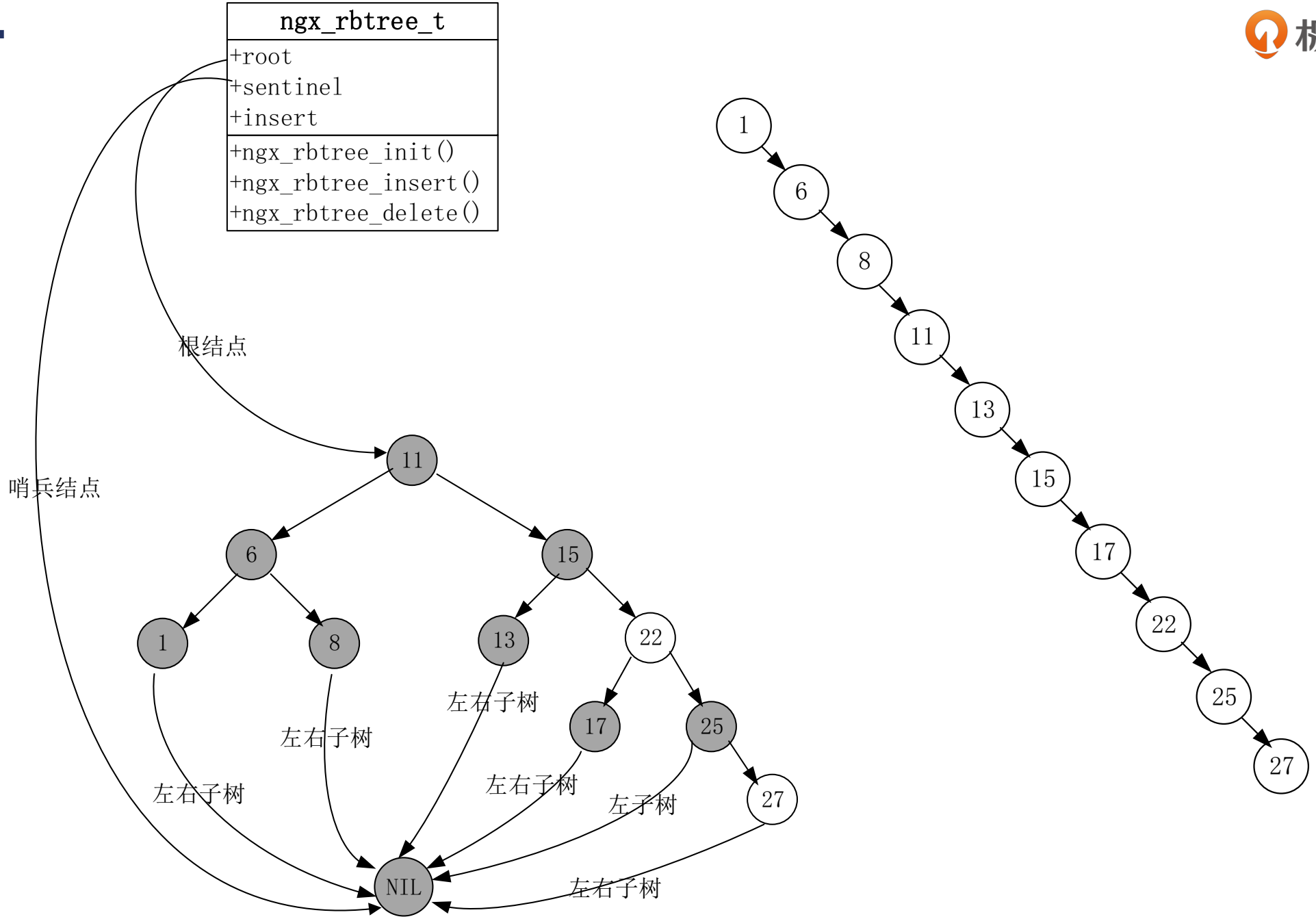
Bucket size

对齐问题

Max size



红黑树


ngx_rbtree_t
+root
+sentinel
+insert
+ngx_rbtree_init()
+ngx_rbtree_insert()
+ngx_rbtree_delete()
根结点
哨兵结点
11
6
15
1
8
13
22
17
25
27
NIL
左右子树
左右子树
左右子树
左右子树
左子树
左右子树
1
6
8
11
13
15
17
22
25
27

红黑树

极客时间

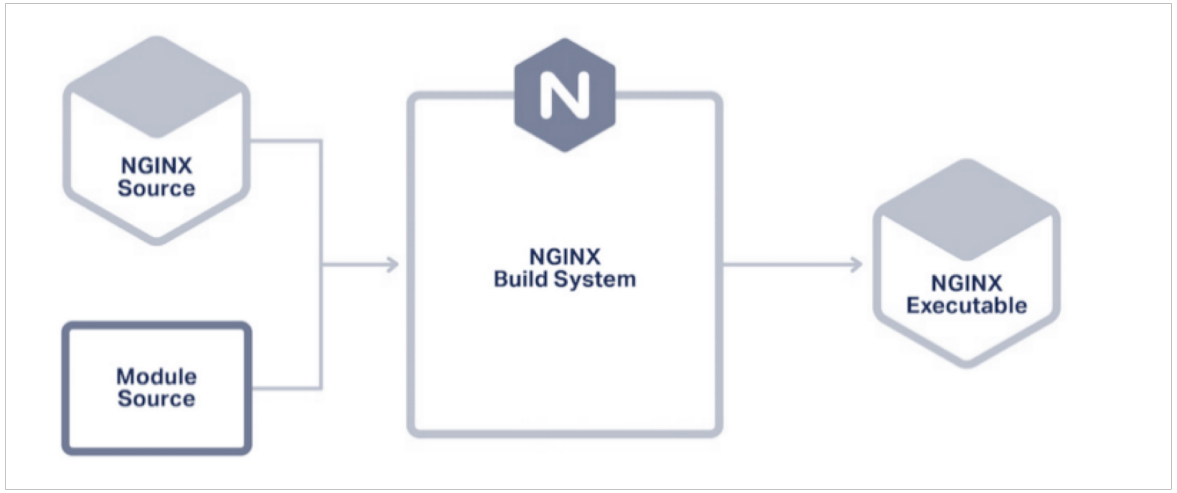
红黑树的使用模块

极客时间

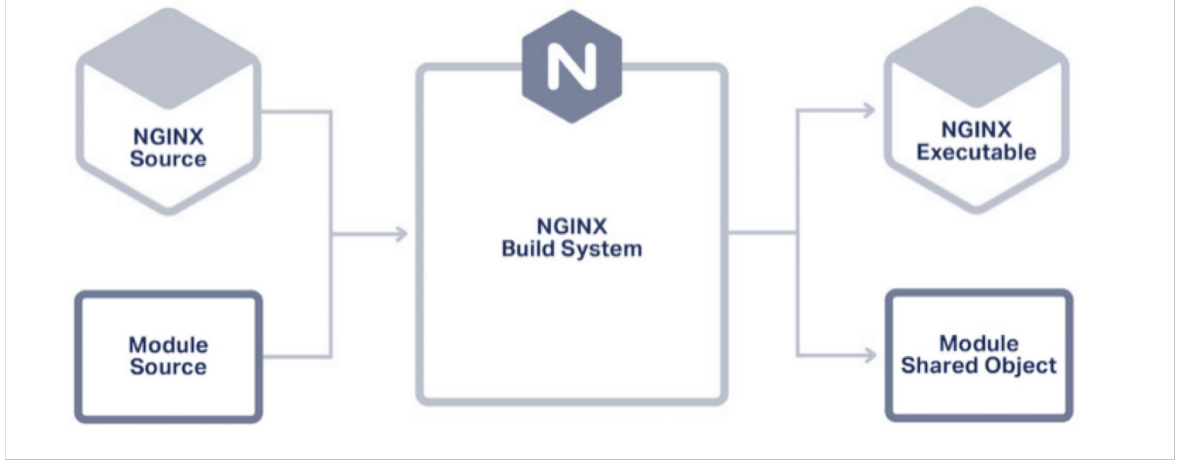
红黑树

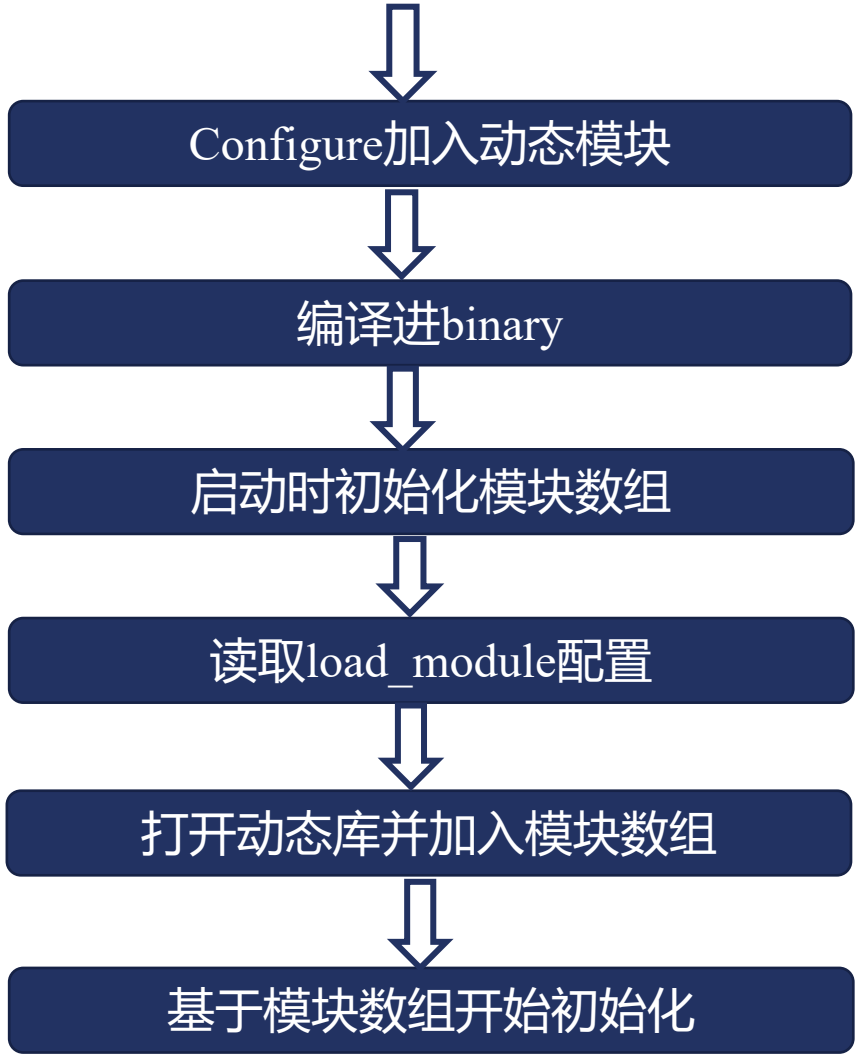
- ngx_conf_module　config_dump_rbtree
- ngx_event_timer_rbtree
- Ngx_http_file_cache
- Ngx_http_geo_module
- Ngx_http_limit_conn_module
- Ngx_http_limit_req_module
- Ngx_http_lua_shdict:ngx.shared.DICT　LRU链表性质
- resolver　ngx_resolver_t
- Ngx_stream_geo_module
- Ngx_stream_limit_conn_module

动态模块-减少编译环节



load_module modules/ngx_http_image_filter_module.so;





极客时间

扫码试看/订阅

《Nginx 核心知识100讲》