

# nginx的实现机制&内部实践

by chenjansen

# About Me

- 陈建森
  - hi : shevacjs
  - 2011/07 入职百度
  - 贴吧技术部 : arch组
  - 百度nginx社区技术负责人

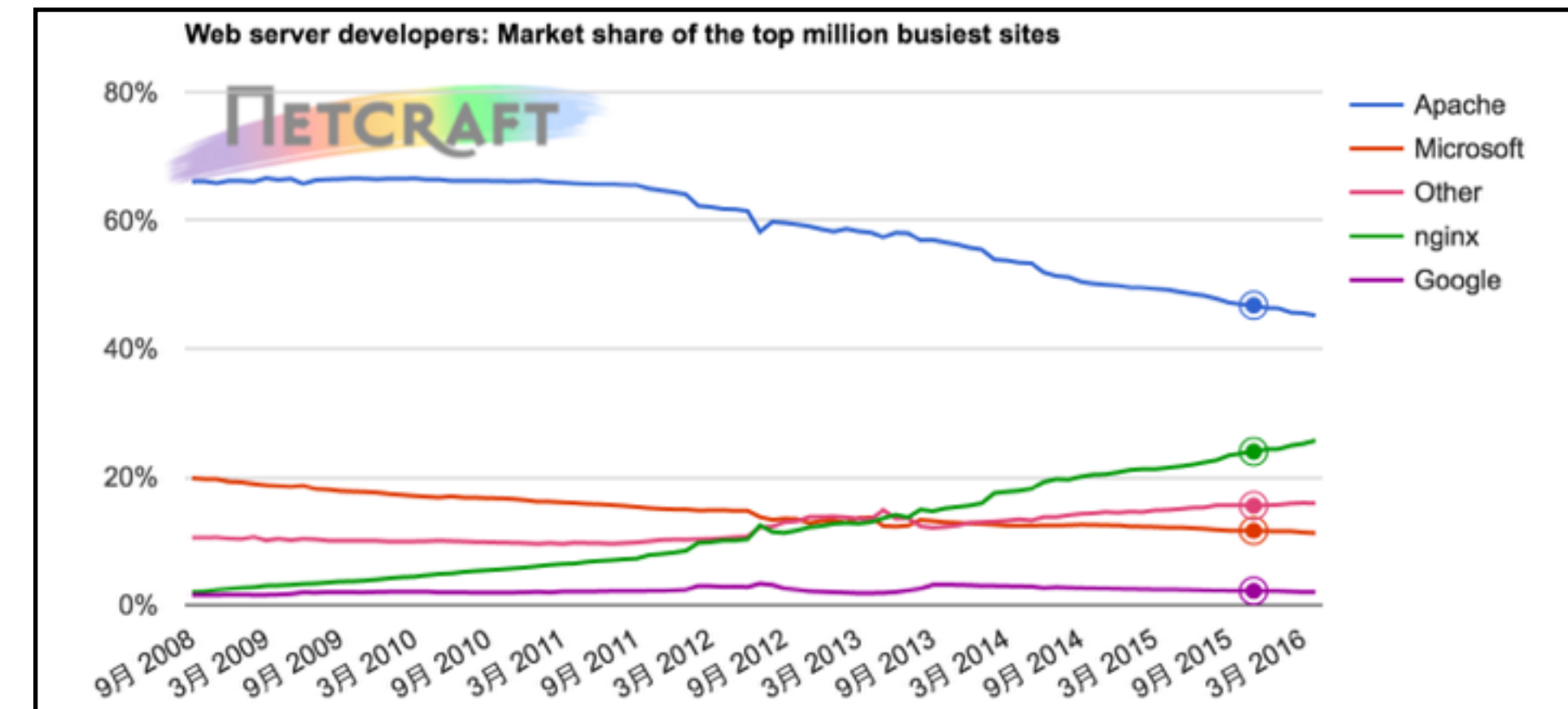
# 大纲

- nginx简介
- nginx核心机制
- nginx扩展开发说明
- nginx在百度的应用实践
- 其他

The NGINX logo is displayed in a bold, green, sans-serif font. The letters are stylized, with the 'i' in 'NGiNX' having a unique shape. The logo is centered horizontally on the right side of the slide.

# nginx简介

- 概述
  - 使用最为广泛的webserver之一
  - 支持 : HTTP 反向/正向代理
  - 目前也支持 : TCP/UDP 层代理
  - 不常用 : 邮箱代理服务器(pop3/smtp)



from [April 2016 Web Server Survey](#)

# nginx简介: 特点

## ☑ 功能强大

- ☑ webserver/ cache 支持
- ☑ keepalive/pipelined 支持
- ☑ 多upstream支持(http, fastcgi)
- ☑ 输出灵活控制 : chunk/gzip
- ☑ 持续发展: http2/ tcp,udp proxy

## ☑ 性能优异

- ☑ 多worker支持
- ☑ 非阻塞网络IO/高并发
- ☑ thread pool 支持文件IO
- ☑ 基于rbtree的定时器
- ☑ 系统特性持续支持

# nginx简介: 特点(续)

## ☑ 运维友好

- ☑ 配置相对友好
- ☑ 支持热加载, 热更新
- ☑ 多server支持
- ☑ 容错性好
- ☑ 日志等自定义支持

## ☑ 扩展方便

- ☑ 代码质量高, 可读性好
- ☑ 社区活跃
- ☑ 状态机模型
- ☑ 高度模块化
- ☑ 丰富的基础组件

# nginx对比

表 1-2 Nginx 与 Apache、Lighttpd 的综合对比

Web 服务器	Nginx	Apache	Lighttpd
反向代理	非常好	好	一般
Rewrite 规则	非常好	好	一般
FastCGI	好	差	非常好
热部署	支持	不支持	不支持
系统压力比较	很小	小	很大
稳定性	非常好	好	一般
安全性	一般	好	一般
技术资料	很少	非常多	一般
静态文件处理	非常好	一般	好
虚拟主机	支持	支持	支持
内存消耗	非常小	很大	非常小

# 如何做到？



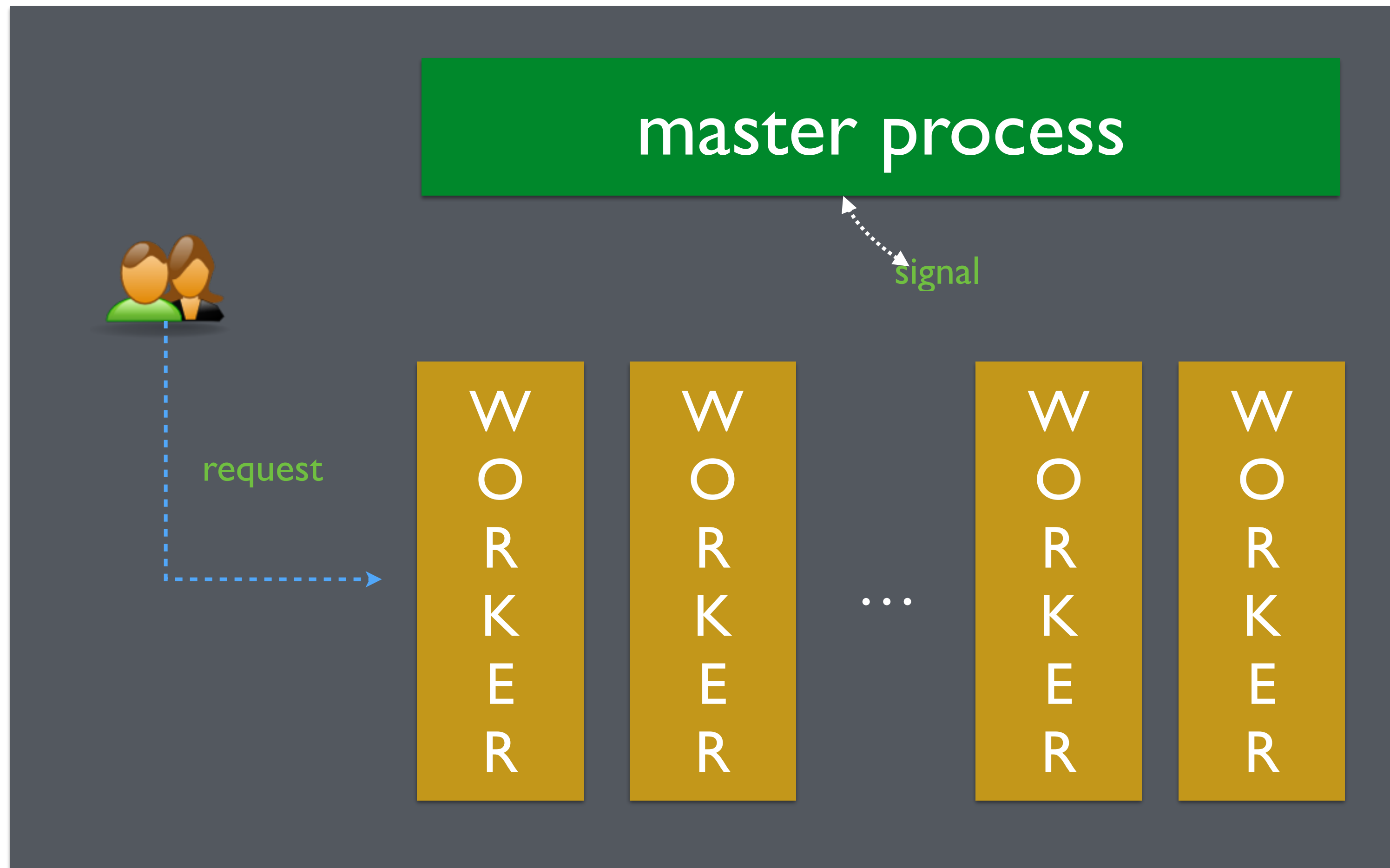


# 大纲

- nginx简介
- **nginx核心机制**
- nginx扩展开发说明
- nginx在百度的应用实践
- 其他

The NGINX logo is displayed in a bold, green, sans-serif font. The letters are stylized, with the 'i' in 'NGiNX' having a unique shape. The logo is centered horizontally on the right side of the slide.

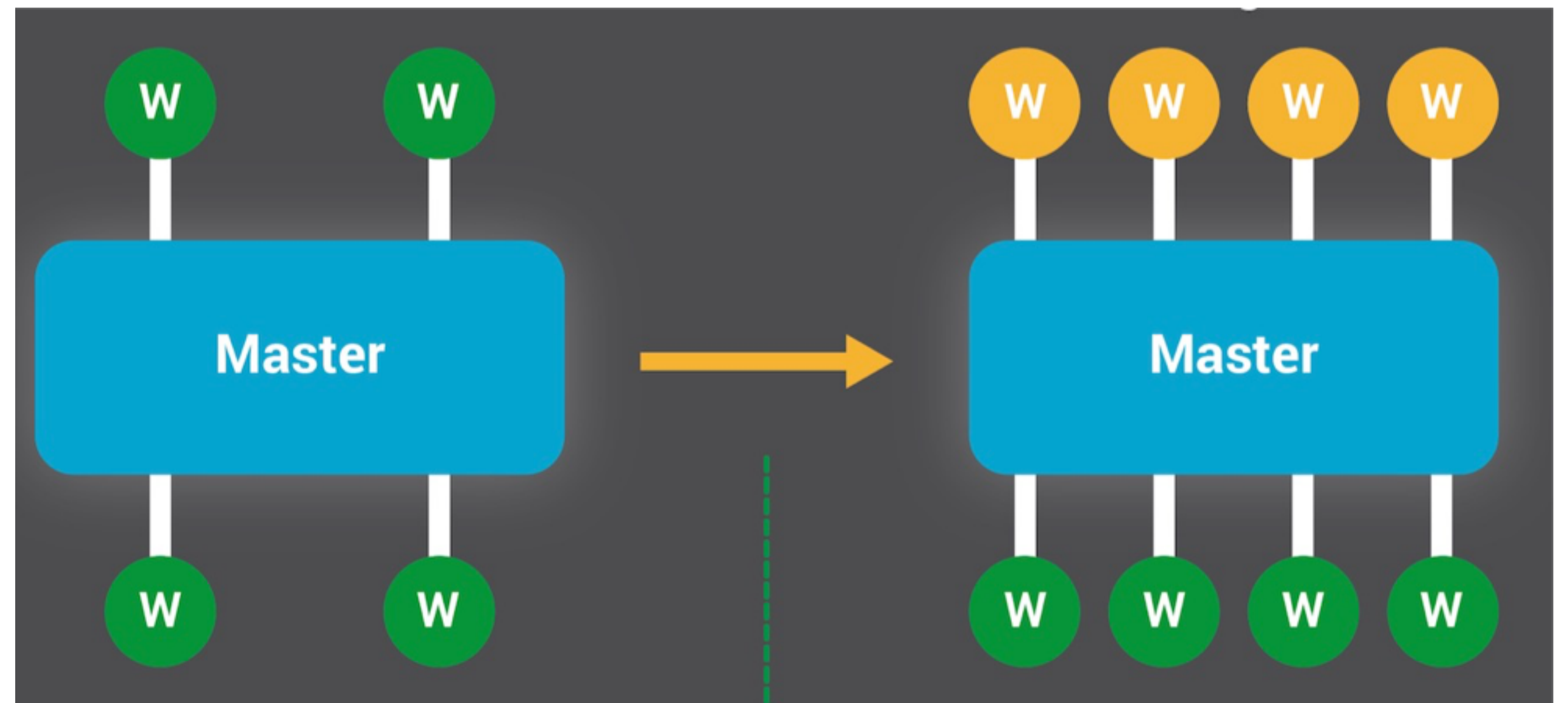
# nginx核心机制: 框架



- 多进程
  - master-worker 模式
  - 单线程/非阻塞

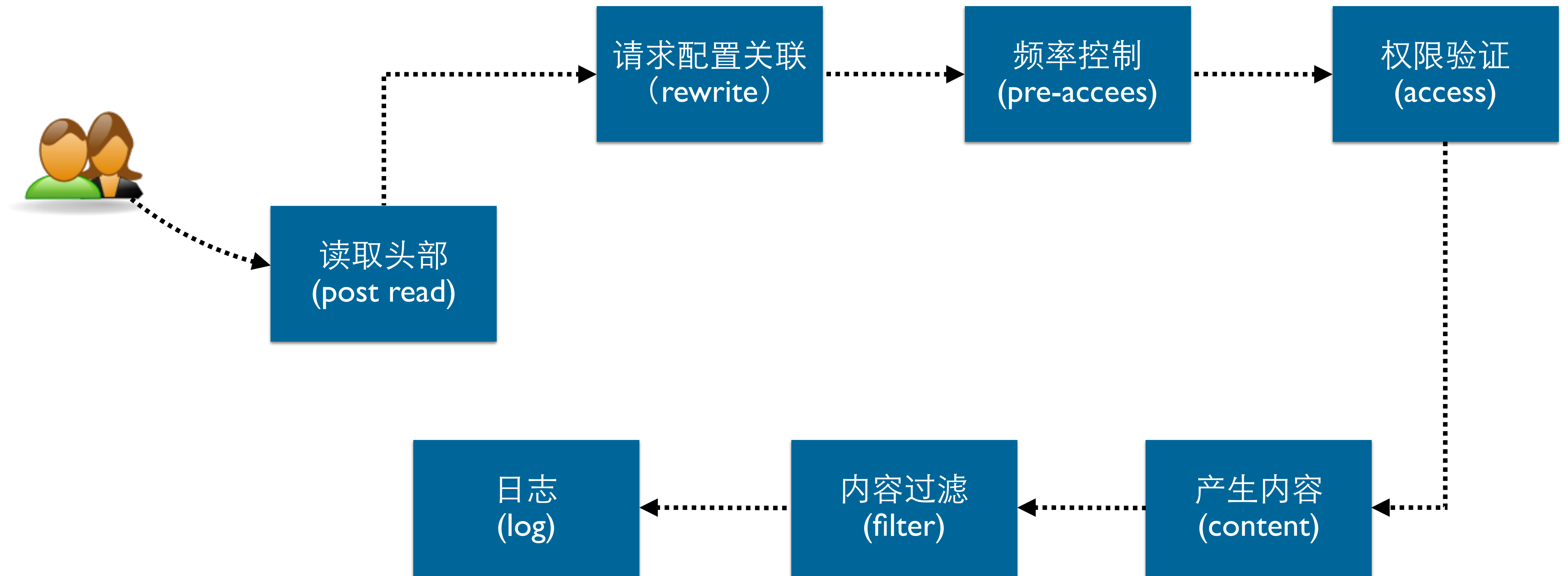
# nginx核心机制: 进程管理

- master : 控制worker
  - 热加载/热更新
  - by signal
- worker : 处理请求

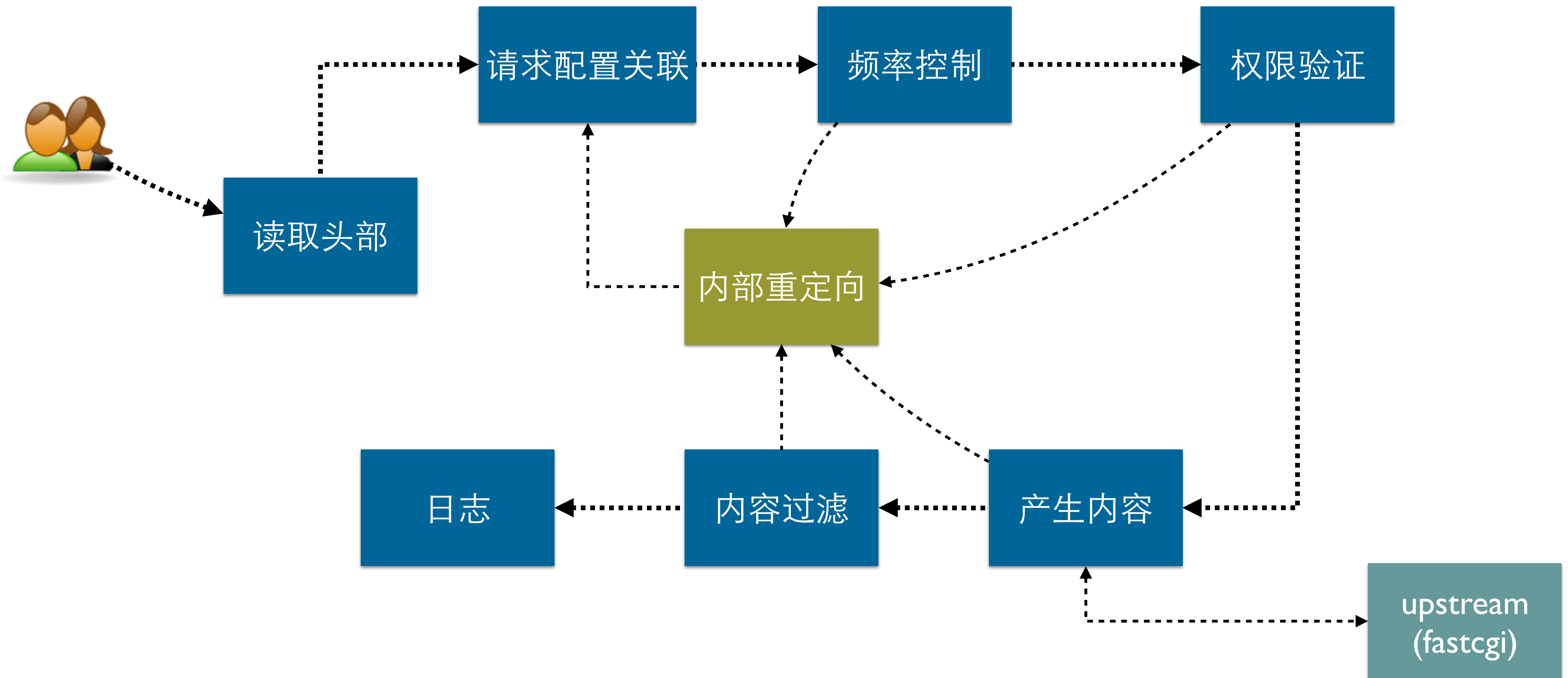


配置热加载示意图: from [Inside Nginx](#)

# worker : 单个请求处理流程



# worker : 单个请求处理流程



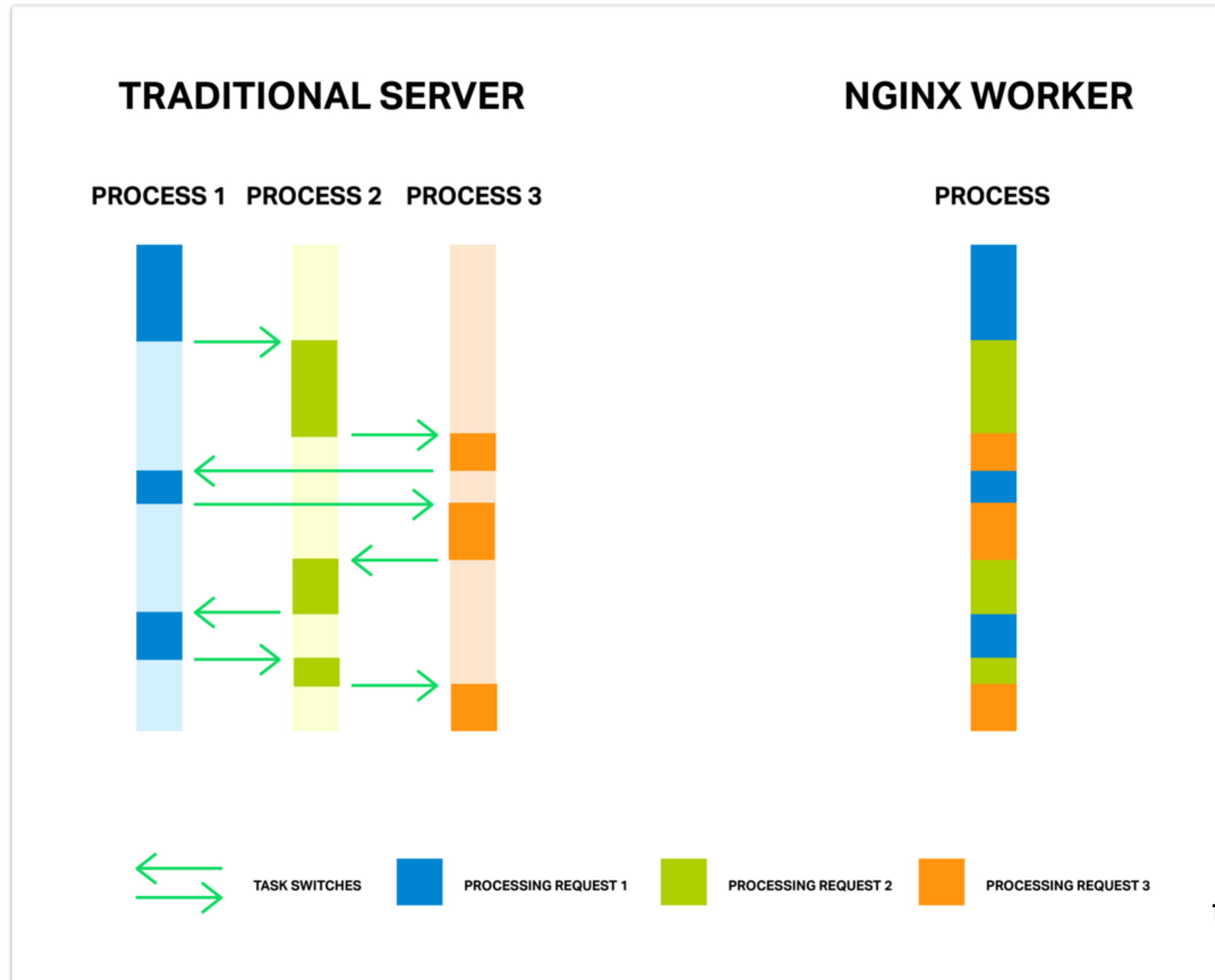
# worker：处理流程状态机

阶段	常见作用
NGX_HTTP_POST_READ_PHASE	任何钩子都可以
NGX_HTTP_SERVER_REWRITE_PHASE	server URL段重写
NGX_HTTP_REWRITE_PHASE	URL重写
NGX_HTTP_PREACCESS_PHASE	速度控制等
NGX_HTTP_ACCESS_PHASE	权限控制
NGX_HTTP_CONTENT_PHASE	内容控制
FILTER	返回包体过滤
NGX_HTTP_LOG_PHASE	日志打印控制

# worker : 请求管理

- 如上说到的单个请求的处理流程
- 并发请求是如何处理的?
  - one request per thread/process ?
  - multi request per process !

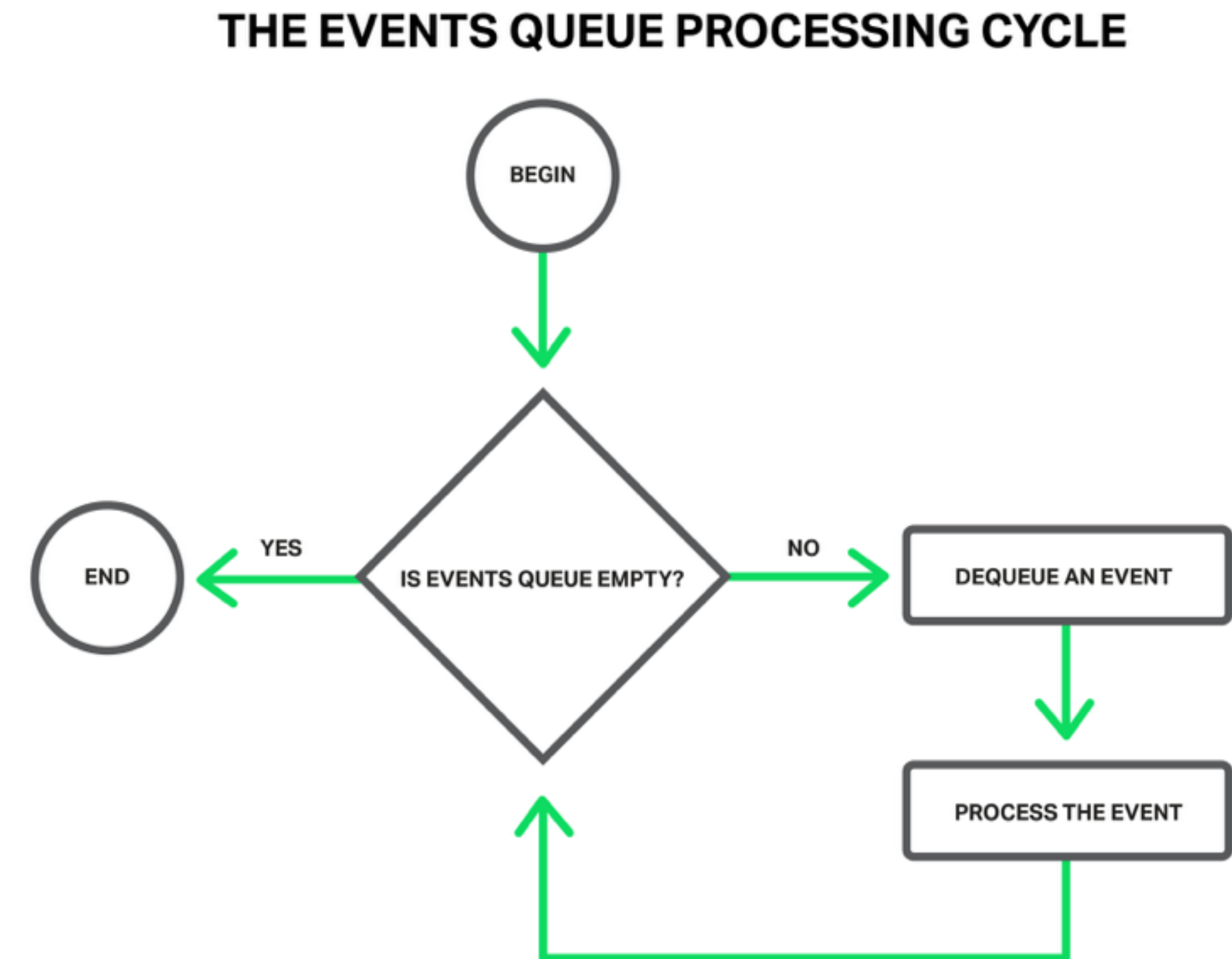
# worker : 请求管理





# 请求管理: how

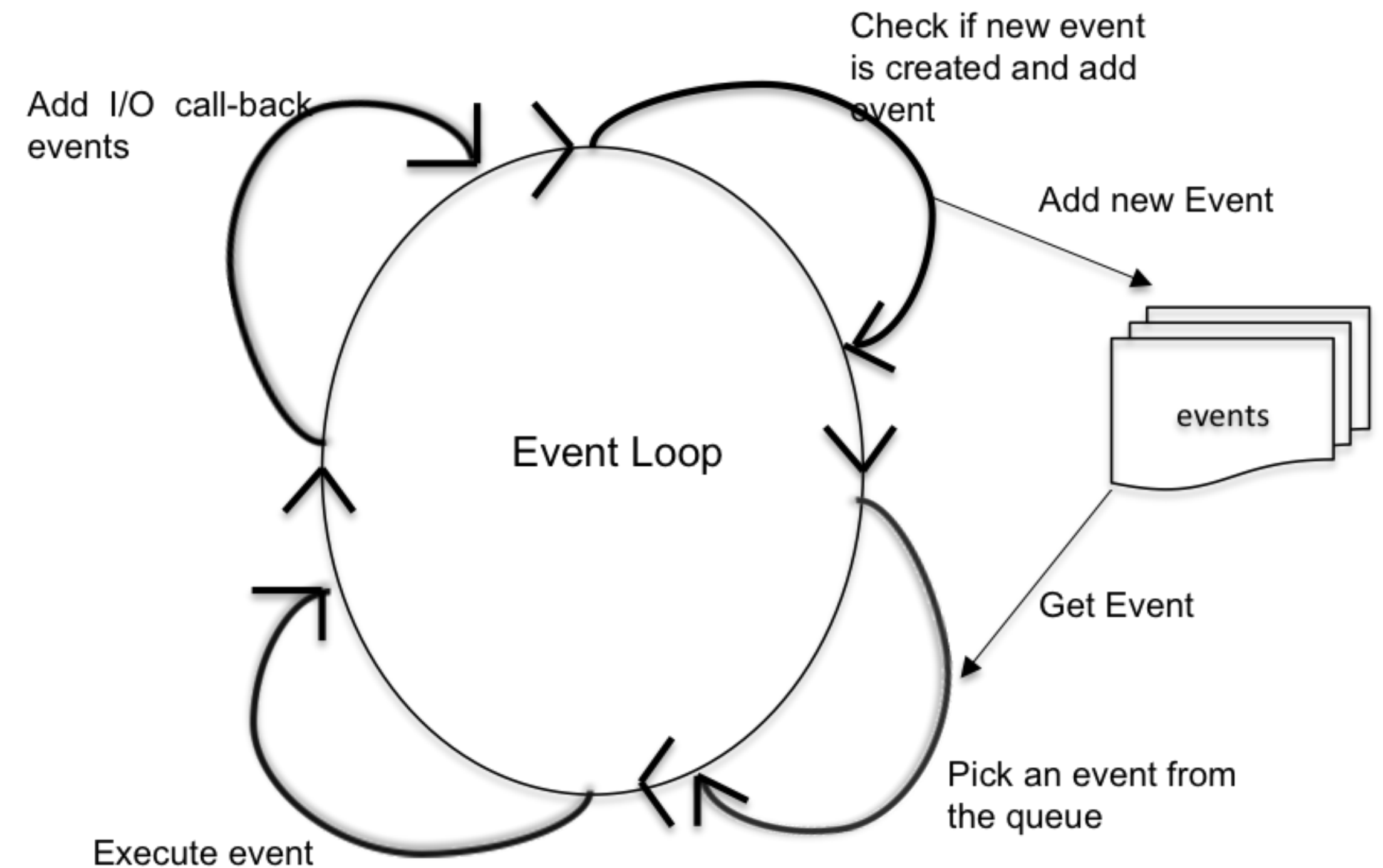
- 基于事件框架的处理模型
  - 请求的调度基于事件
  - 事件
    - 网络IO(可读/可写)
    - 定时器
    - 磁盘IO



from [Thread Pools in NGINX](#)

# 事件管理: 网络IO

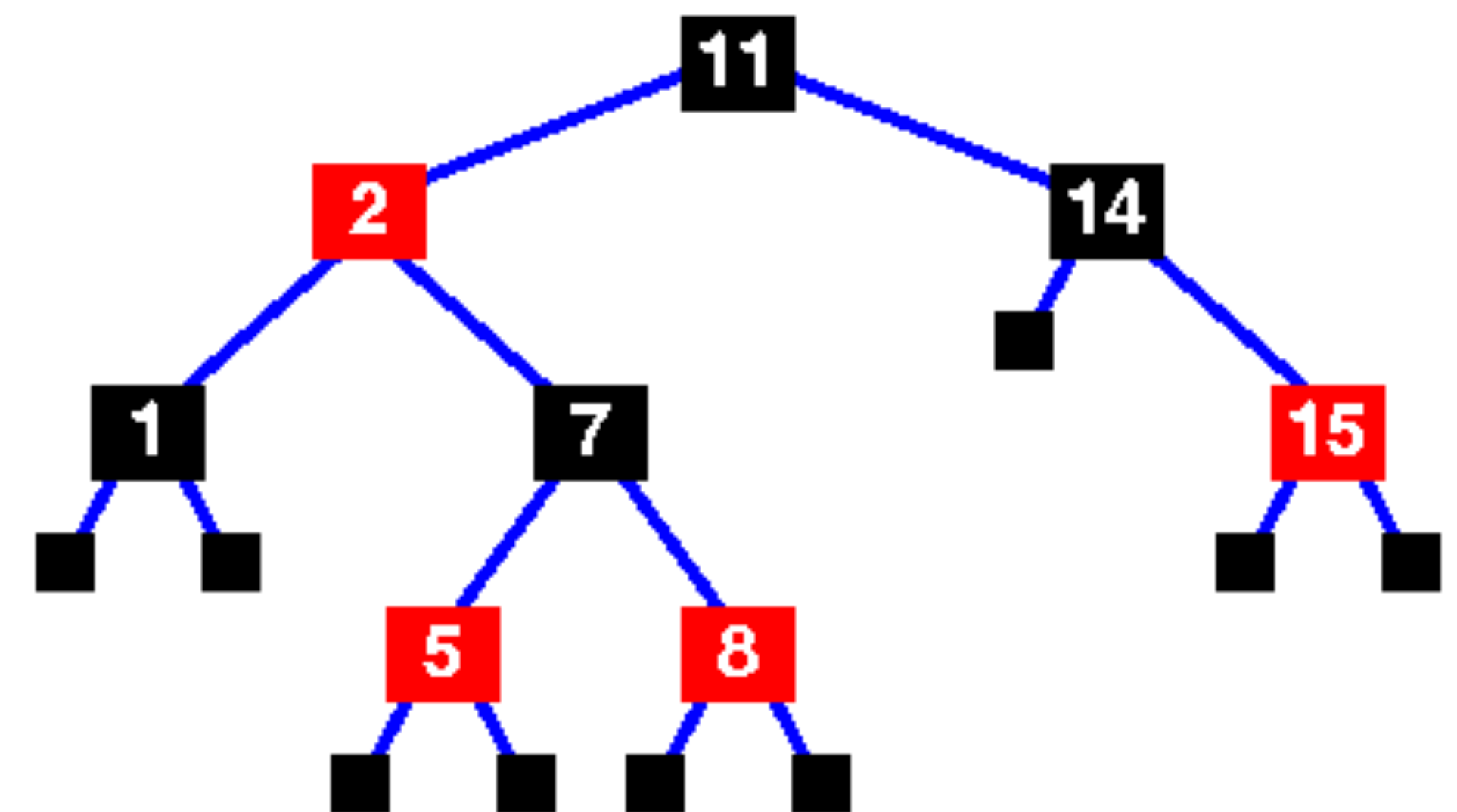
- 网络IO管理
  - 基于epoll的事件分发(linux下)
  - 所有IO非阻塞
  - 类似于reactor模式



from [Reactor Patter2](#)

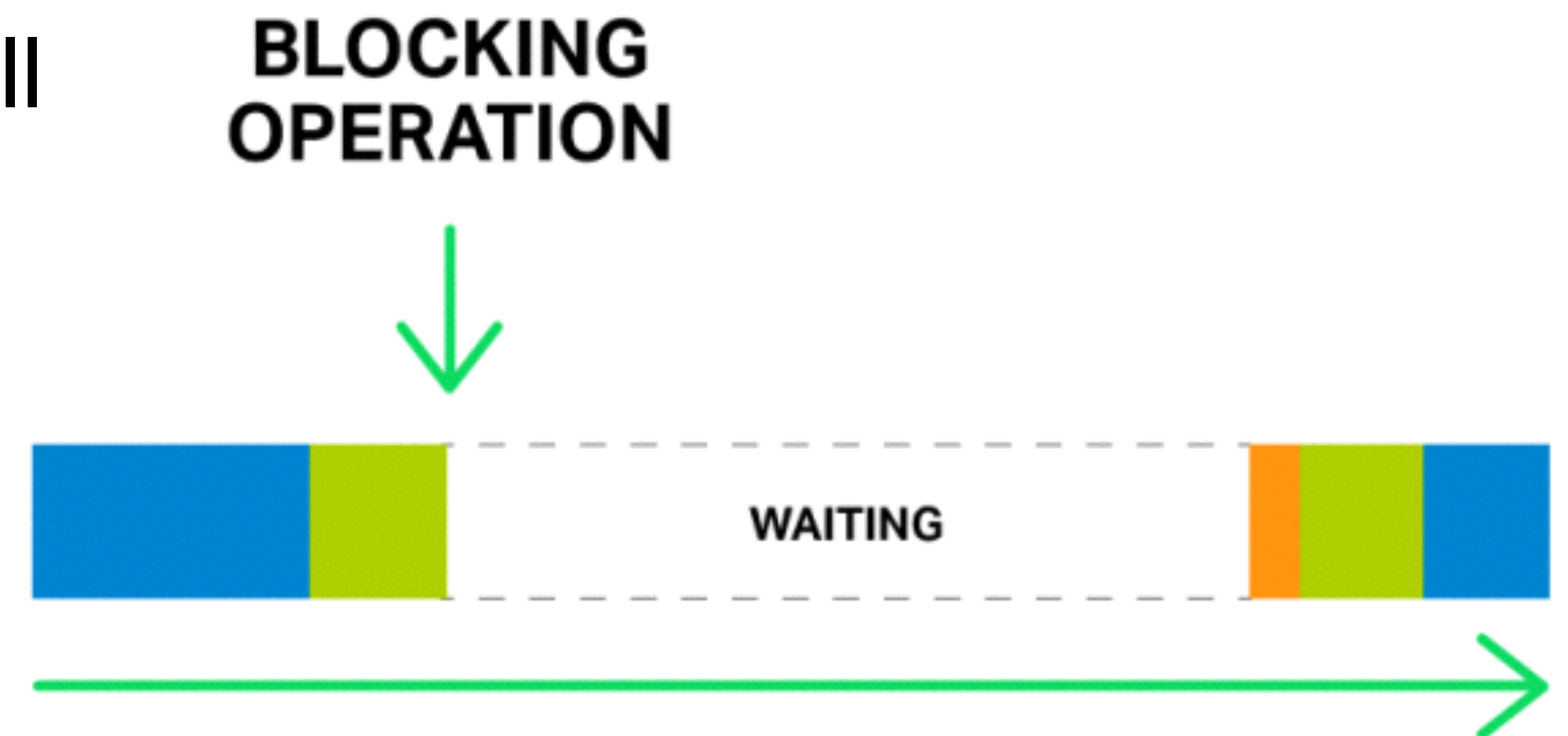
# 事件管理: 定时器

- 定时器
  - 作用: 定时任务/超时管理
  - 要求: 快速查找/插入
  - 实现: 基于rbtree
    - 最左边节点即最近超时



# 事件管理: 磁盘IO

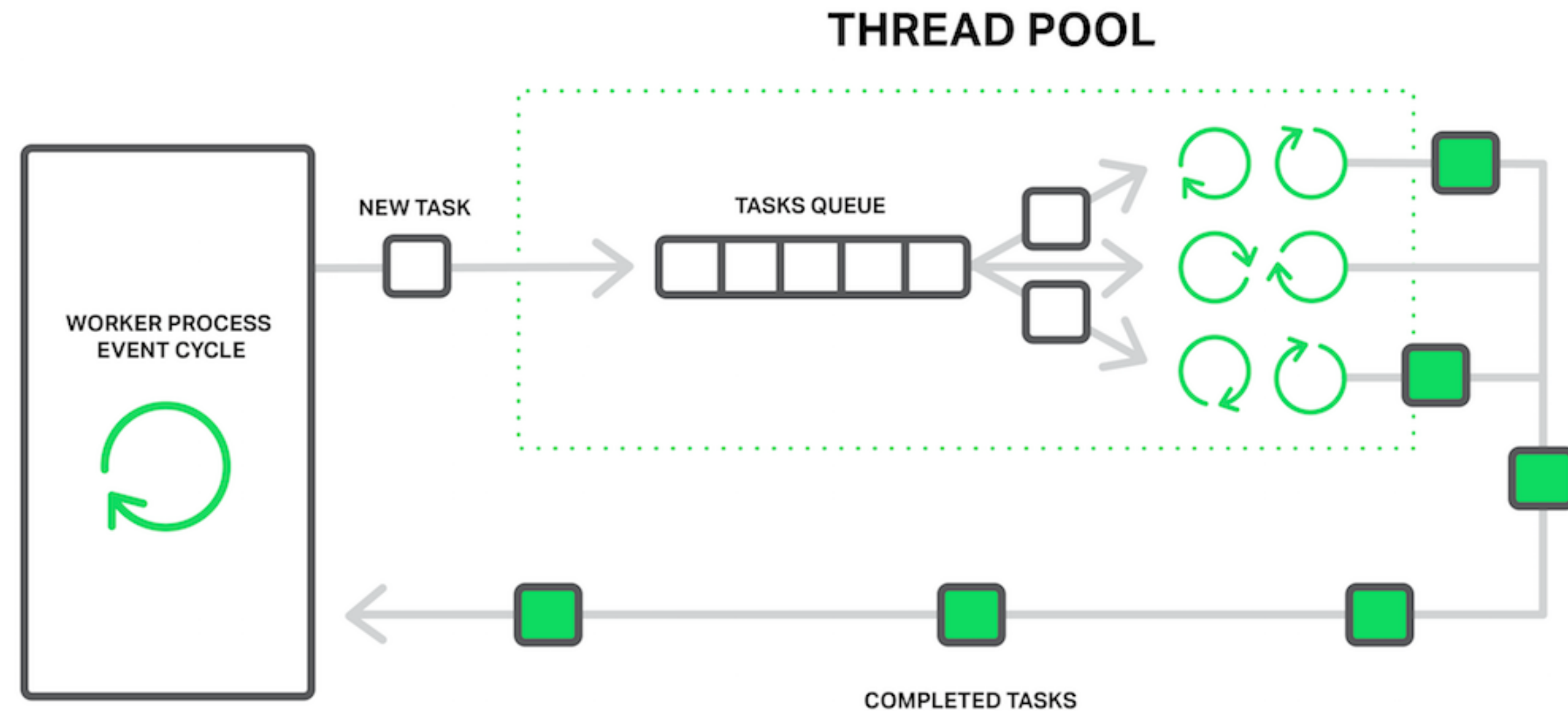
- 磁盘IO管理: 大文件读写
  - Linux的异步IO机制(AIO) & epoll
  - 问题:
    - AIO必须打开IO\_DIRECT
    - AIO会关闭sendfile
- 新方案: thread pool



from [Thread Pools in NGINX](#)

# 事件管理: 磁盘IO

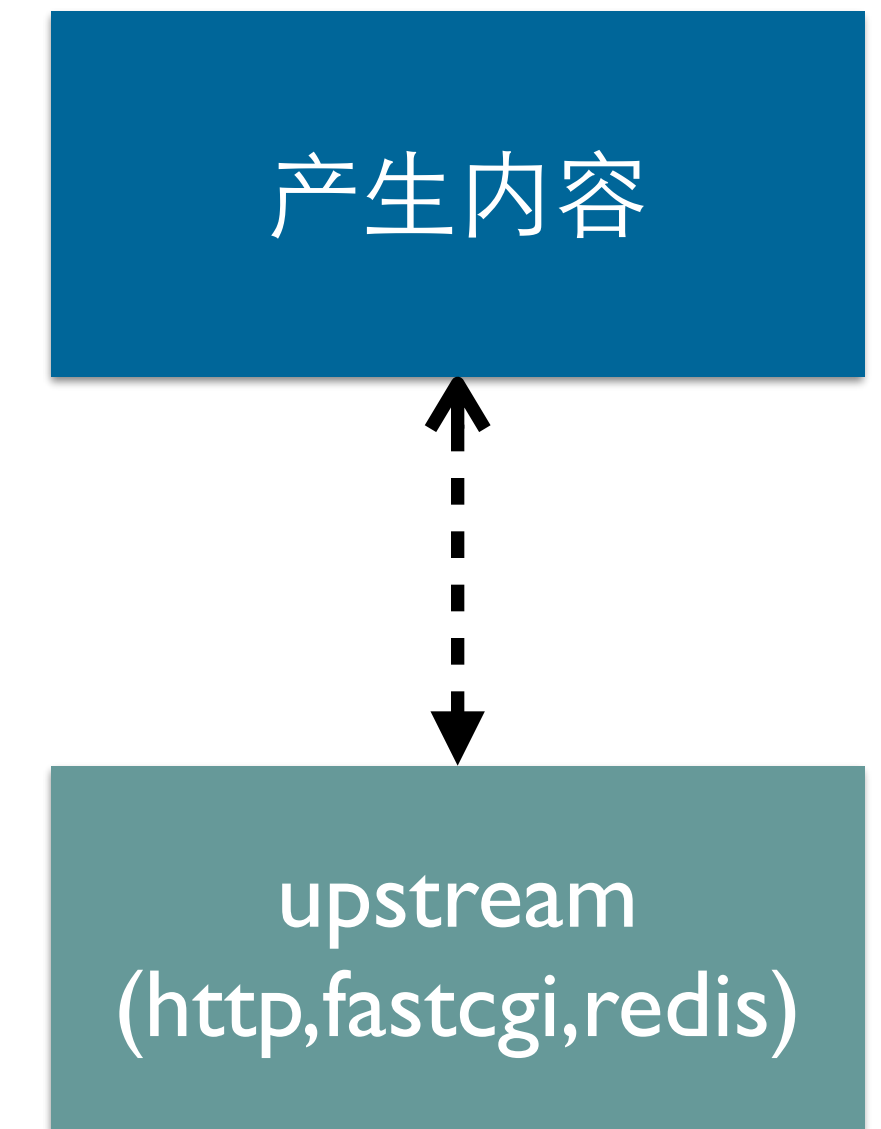
- 磁盘IO管理: 大文件读写
  - thread pool & epoll
  - 单独线程用于IO的读写
  - 基于pthread



from [Thread Pools in NGINX](#)

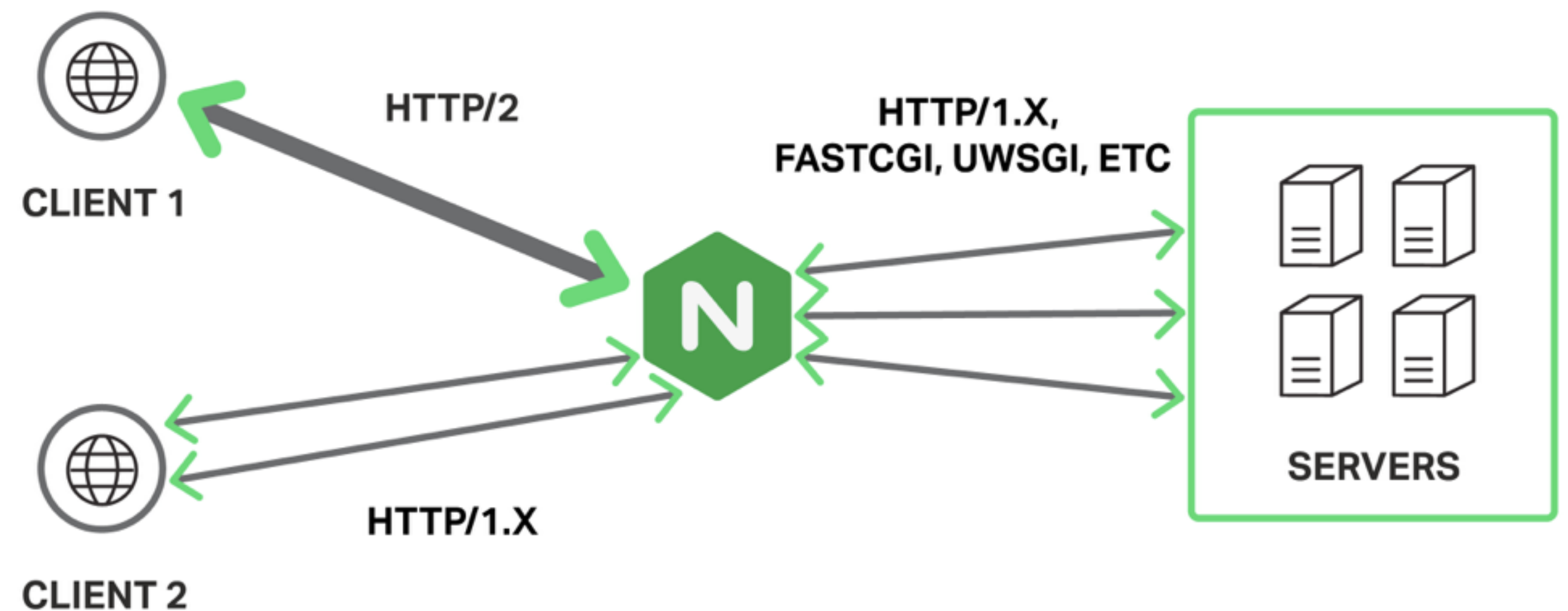
# nginx核心机制: upstream

- 内容产生PHASE
  - 基本：单机文件读取返回
  - 扩展：与其他upstream交互返回内容
    - 扩展性大大增强
    - HTTP代理/Fastcgi代理/Memcache代理 etc



# nginx核心机制: upstream

- 关键问题
  - 选择哪一个 upstream
  - 如何进行交互
  - 性能上面的优化点





# nginx核心机制: upstream

- upstream选举: 选举一个适当的upstream
  - 资源定位:
    - DNS : resolve 支持
    - 配置变更: dynamic upstream/ bns module
    - 异常识别: fail\_timeout+max\_fails/health\_check
  - 负载均衡: ip\_hash/keepalive/least\_conn/hash etc.
  - 重试控制: proxy\_next\_upstream\_tries etc.

```
upstream dynamic {
    zone upstream_dynamic 64k;

    server backend1.example.com weight=5;
    server backend2.example.com:8080
fail_timeout=5s slow_start=30s;
    server 192.0.2.1 max_fails=3;
    server backend3.example.com resolve;
    server backend4.example.com service=http
    resolve;

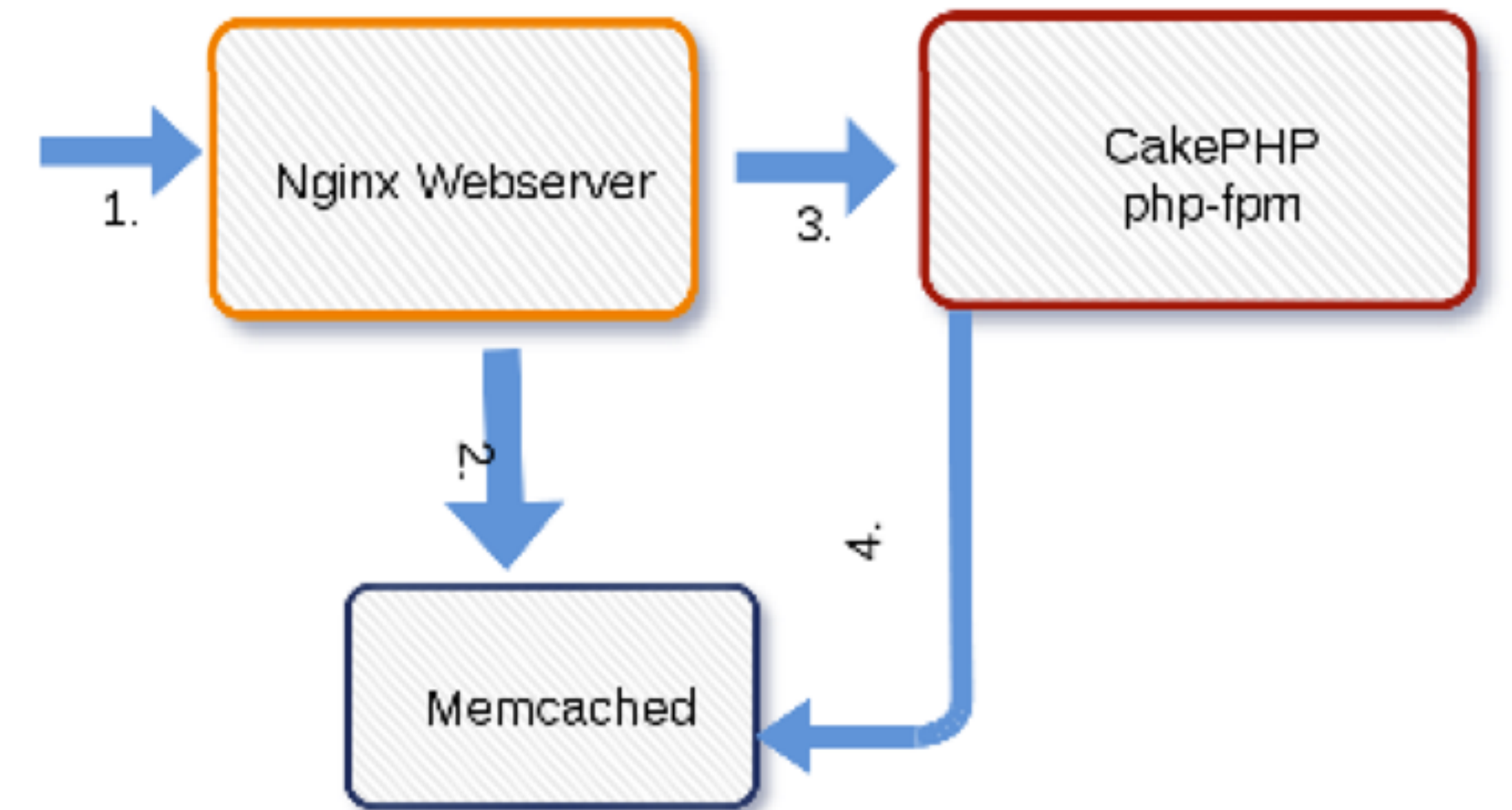
    server backup1.example.com:8080 backup;
    server backup2.example.com:8080 backup;
}

server {
    location / {
        proxy_pass http://dynamic;
        health_check;
    }
}
```



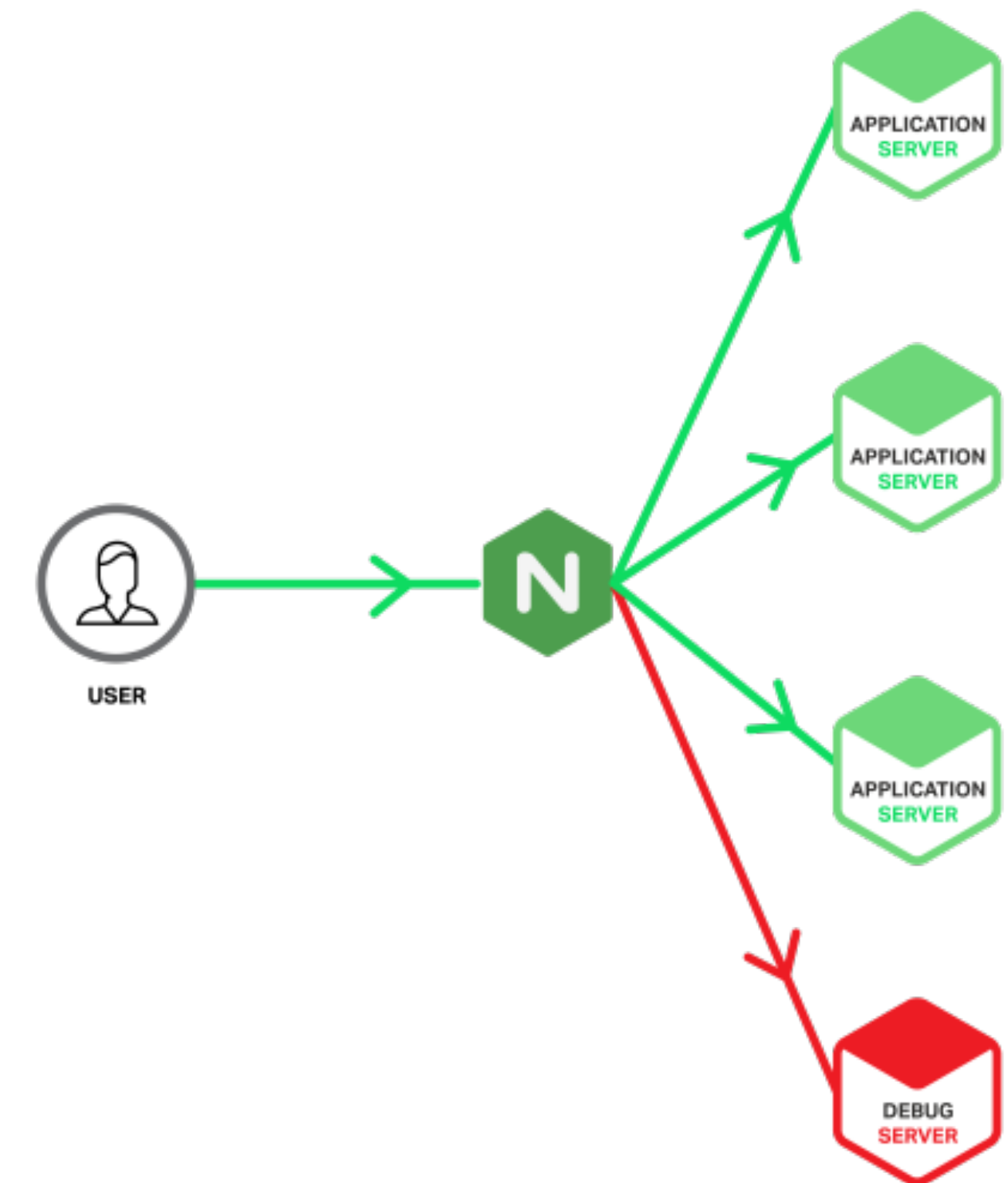
# nginx核心机制: upstream

- 和upstream进行交互
  - 支持基于TCP的任何协议
  - 业内：http/memcached/redis/mysql
  - 公司内：mcgi 协议
  - 实践case
    - 贴吧基于memcached的nginx cache



# nginx核心机制: upstream

- upstream交互优化点: buffering 策略
  - 问题: client & upstream 速度的不一致
  - buffering开启: client读取慢情况
  - buffering关闭: client快,直接返回
  - 实践case: 贴吧的bigpipe迁移



# 核心机制: 其他

- nginx对性能的极限追求，衍生很多机制，包括
  - accept mutex解决惊群效应
  - 支持比如SO\_REUSEPORT特性(1.9.1)
  - time resolution减少系统调用
  - ngx\_buf\_t/ngx\_chain\_t引入保证内存zero-copy
  - 内核参数的持续支持 : tcp\_cork / tcp\_defer\_accept
  - 自定义的内存管理机制

# 大纲

- nginx简介
- nginx核心机制
- **nginx扩展说明**
- nginx在百度的应用实践
- 其他

The NGINX logo is displayed in a bold, green, sans-serif font. The letters are thick and blocky, with a slight shadow effect. The 'i' in 'NGINX' has a unique design with a dot that is a small circle.

# nginx扩展说明

- 关于Nginx扩展： 关键特性
  - 整体框架设计支持(状态机模型 etc.)
  - 所有功能都以此类方式提供
  - 1.9.11 版本开支支持 dynamic module
  - 构建强大的生态
    - nginx\_lua\_module、 nginx\_rtmp\_module
    - 单 [nginx 3rd party module](#) 页面就记录130个扩展

# nginx扩展说明

- 分类
  - 按照对应子系统：http、stream、event
  - 按照hook阶段：处理模块/过滤模块/负载均衡模块
  - 按照语言：C/C++/Lua/JavaScript
- 大部分扩展
  - 基于C/lua的http扩展

# nginx扩展开发Demo

- Nginx扩展开发demo
  - 以nginx\_realip\_module为例
  - Http的处理模块的C扩展
  - 关键步骤
    - 模块注册、钩子注册/实现、编译加载



# nginx扩展开发Demo: 模块注册

- 模块注册
  - 指令定义
  - 启动钩子定义
  - 变量定义
  - 模块定义

```
static ngx_command_t  ngx_http_realip_commands[] = {  
    { ngx_string("set_real_ip_from"), → 指令名  
      NGX_HTTP_MODULE, NGX_COMMANDS_SINGLE, NULL, 0 },  
    ngx_null_command  
};  
  
ngx_module_t  ngx_http_realip_module = {  
    NGX_MODULE_V1,  
    &ngx_http_realip_module_ctx,      /* module context */  
    ngx_http_realip_commands,         /* module directives */  
    NGX_HTTP_MODULE,                  /* module type */  
    NULL,                              /* init master */  
    NULL,                              /* init module */  
    NULL,                              /* init process */  
    NULL,                              /* init thread */  
    NULL,                              /* exit thread */  
    NULL,                              /* exit process */  
    NULL,                              /* exit master */  
    NGX_MODULE_V1_PADDING  
};  
  
ngx_null_command
```

模块定义: 类型, CTX



# nginx扩展开发Demo: 钩子注册

- 钩子控制
  - 钩子注册
  - 钩子实现

```
static ngx_int_t
ngx_http_realip_init(ngx_conf_t *cf)
{
    static ngx_int_t
    ngx_http_realip_handler(ngx_http_request_t *r)
    {
        // 代码有删减 具体回调函数实现

        ctx = ngx_http_get_module_ctx(r, ngx_http_realip_module);

        if (ctx) {
            return NGX_DECLINED;
        }

        rlcfcf = ngx_http_get_module_loc_conf(r, ngx_http_realip_module);

        if (rlcfcf->from == NULL) {
            return NGX_DECLINED;
        }

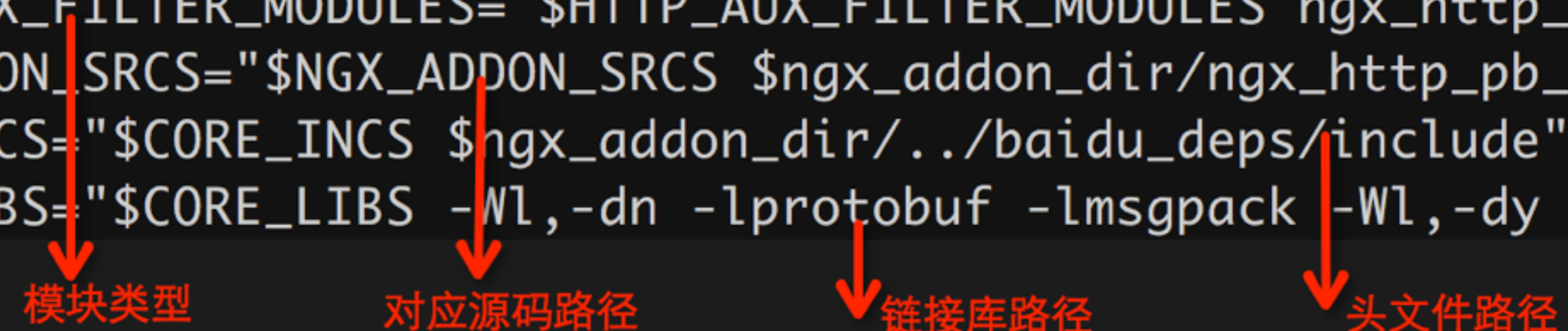
        // 代码有删减

        switch (rlcfcf->type) {
```

# nginx扩展开发Demo: 编译加载

- 控制如何编译加载
  - 静态/动态模块
  - 编写config文件

```
ngx_addon_name=ngx_http_pb_msgpack_transfer_module
HTTP_AUX_FILTER_MODULES="$HTTP_AUX_FILTER_MODULES ngx_http_pb_
NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_pb_msg
CORE_INCS="$CORE_INCS $ngx_addon_dir/../../baidu_deps/include"
CORE_LIBS="$CORE_LIBS -Wl,-dn -lprotobuf -lmsgpack -Wl,-dy -L$
```



模块类型      对应源码路径      链接库路径      头文件路径

# nginx扩展说明

- 问题&未来
  - 传统C/C++扩展对迭代效率/稳定性的影响
  - nginx/lua 在性能/稳定性/效率 方面的折衷
    - 简单复杂需求的理想方案/ 衍生出来的openresty生态
- 未来：lua化
  - nginx/lua大规模开发挑战：规范化/标准化
  - 更多的lua C扩展 or nginx-lua C 扩展

# 大纲

- nginx简介
- nginx核心机制
- nginx扩展开发说明
- **nginx在百度的应用实践**
- 其他

The NGINX logo is displayed in a bold, green, sans-serif font. The letters are stylized, with the 'i' and 'n' having unique shapes. The logo is centered horizontally on the right side of the slide.

# nginx在百度

- 公司内外有诸多基于nginx的解决方案
  - Nginx Plus
  - Openresty 项目
  - Tengine 项目
  - 百度内部的实践情况

# nginx在百度

- Nginx Plus : nginx作者创建的商业化版本
  - 基于nginx功能的增强, 比如:
    - 更强的负载均衡策略(动态upstream/流控 etc.)
    - 支持HLS视频流/支持HDS(Adobe HTTP Dynamic Streaming )
    - session log
  - 基于nginx生态的建设
    - 比如: Nginx Amplify

# nginx在百度

- OpenResty：国人章亦春([agentzh](#))主导
  - 目标：web服务直接跑在nginx内部
  - 基于nginx与lua的高性能平台
  - 不修改任何nginx主干代码
  - 含有大量的Lua库，第三方模块，依赖项
  - 外网生态建设良好：自动化功能/性能测试(systemtap script etc.)



# nginx在百度

- Tengine：淘宝网2011年发起的开源项目
  - 目标：打造一个高效, 稳定, 安全, 易用的web平台
  - 基于nginx增加诸多高级特性
  - 对原生nginx核心有些许微调：兼容性问题
  - 一些feature：
    - nginx(nginx plus)逐渐支持：动态模块/动态upstream etc.
    - 暂不支持：命令行优化/系统过载



# nginx在百度

- 发展历程
  - 贴吧
    - 2010年开始试用静态文件
    - 2012年所有webserver迁移nginx
- 其他
  - PS/know/wise也相继迁移nginx
  - 随着ODP的大面积推广

# nginx在百度

- 开发实践
  - 原则上不修改源码
  - 专注于
    - 扩展开发：支持业务的快速发展
    - 内核特性研究：保证问题/性能等问题的解决

# nginx在百度: 现状

- 基础扩展支持

- ✓ 3.1. 防攻击模块
- ✓ 3.2. 新防攻击模块
- ✓ 3.3. 设备识别模块
- ✓ 3.4. 问题定位模块
- ✓ 3.5. BAIDUID生成模块
- ✓ 3.6. KETAMA HASH模块
- ✓ 3.7. BNS扩展支持模块
- ✓ 3.8. MCGI协议
- ✓ 3.9. 按组QUERY\_HASH调度

- 内核特性研究

- ✓ 2.1. NGINX 下的异步文件IO管理
- ✓ 2.2. 版本追踪: 1.9.9
- ✓ 2.3. 版本追踪: 1.9.10
- ✓ 2.4. 版本追踪: 1.9.13
- ✓ 2.5. NGINX VARIABLE FLAG说明
- ✓ 2.6. TCP PROXY说明
- ✓ 2.7. NGINX TCP KEEPALIVE PROXY说明

具体可以参见[内部官网](#)

# nginx在百度

- Nginx开发者社区(hi群号: 1509031)
  - 年初专门成立
  - 和ODP深度合作
  - 覆盖: 贴吧/商业搜索/云计算/金融事业部 诸多同学
  - 目标: 支持nginx在百度的落地实践

# nginx在百度

- Nginx开发者社区: 近一个Q工作情况
  - nginx realase包管理
    - 支持 nginx 1.10.1
    - 升级nginx\_dev\_kit/upstream eva 等模块
    - gcc环境检查和容错
    - 部分扩展动态化尝试

# nginx在百度

- Nginx开发者社区: 近一个Q工作情况
  - 通用功能支持
    - nginx pagecache扩展开发
    - nginx 通用防攻击模块增强
    - nginx lua开发框架梳理

# nginx在百度: 将来

- nginx研发全流程
  - release包 / 工具生态 etc.
- nginx核心功能支持
  - 基础C扩展支持/ 新扩展开发
- nginx/lua(openresty)开发体系
  - lua开发体系
- 开源：贡献社区



# 大纲

- nginx简介
- nginx核心机制
- nginx扩展开发说明
- nginx在百度的应用实践
- 其他

The NGINX logo is displayed in a bold, green, sans-serif font. The letters are stylized, with the 'i' in 'NGiNX' having a unique shape. The logo is centered horizontally on the right side of the slide.



# 学习建议

- 初步
  - 配置nginx, 了解nginx的基本机制
  - 依照demo, 写一些简单的nginx扩展
- 进阶
  - 了解http/https/http 2协议 / 了解nginx内部机制
- 再进一步
  - 基于优化点的系统深入了解 : sendfile / tcp\_defer\_accept etc.
  - 延伸生态的学习 : luajit / systemtap etc.

# 推荐资料

- 链接
  - [官方模块说明指南](#)
  - [内部模块介绍](#) & [开发指南](#)
  - 淘宝[tengine](#)书籍
  - [openresty git](#) 和 [章亦春blog](#)
- 书籍
  - [深入理解nginx](#)：重点介绍内部实现机制
  - [精通Nginx](#)：偏应用实践

