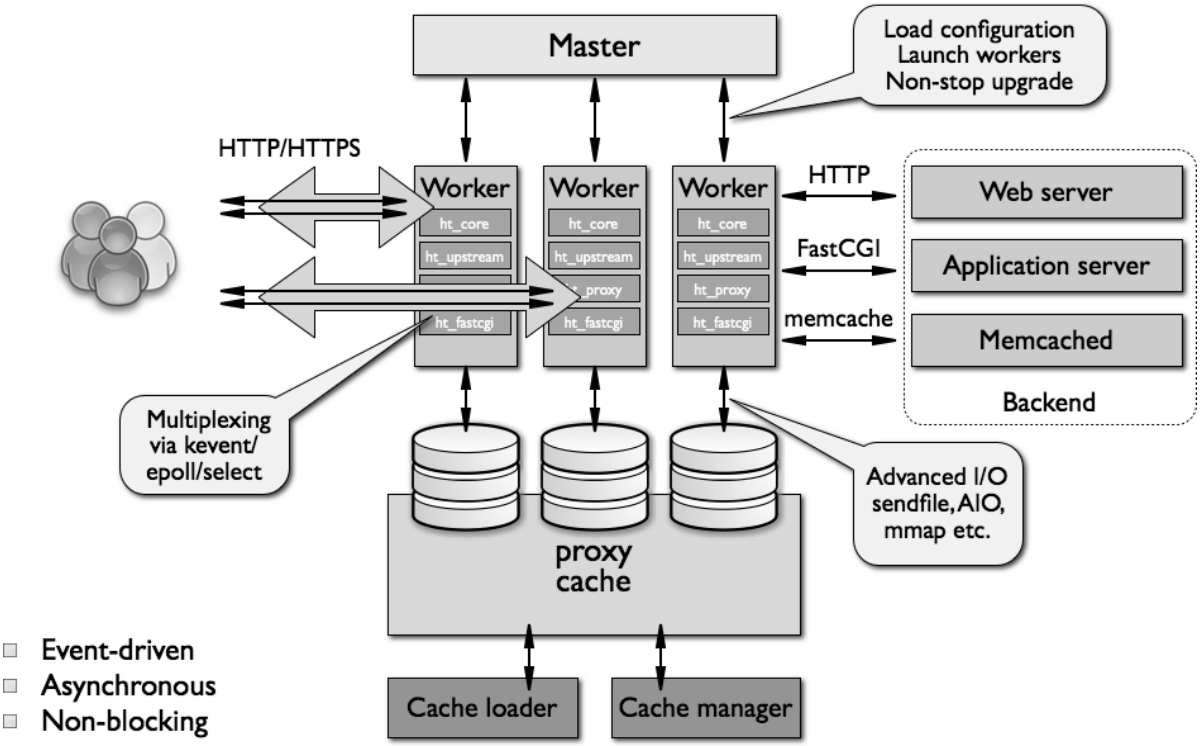
nginx介绍

> nginx最主要的功能是：web服务器、反向代理服务器(负载均衡器)、缓存服务器。
>
> nginx特性：模块化设计。1.9.11以及更新的版本已经支持动态模块加载；高可靠性。主控进程master负责管理子进程worker；内存消耗少；支持不关闭程序更新配置文件、升级程序；支持事件驱动、AIO、MMAP；高并发性；支持rewrite重写；内置健康检查机制。

nginx架构



> master负责生成并管理worker进程，worker进程负责接收客户端请求，通过查找配置文件，将客户端的请求映射到一个location block，location中的不同指令生效启动不同的模块去完成响应请求的工作。同时，worker进程可以基于不用的协议(http、fastcgi等)与后端服务器进程交互，如果nginx作为反向代理服务器，可以提供缓存加速及缓存管理等功能，同时，支持worker进程与磁盘的高性能IO模型。支持事件驱动，默认是边缘触发的通知机制。

编译nginx

> 创建用户组、用户

```
groupadd nginx
useradd -M -g nginx nginx -s /sbin/nologin
```

> 安装依赖库

```
yum install openssl-devel pcre-devel gd-devel
```

> nginx编译参数

```
./configure \
--user=nginx \
--group=nginx \
--prefix=/usr/local/nginx \
--conf-path=/etc/nginx/nginx.conf \
--http-log-path=/var/nginx/access.log \
--error-log-path=/var/nginx/error.log \
--with-http_ssl_module \
--with-http_flv_module \
--with-http_gunzip_module \
--with-http_gzip_static_module \
--with-http_stub_status_module \
--with-http_image_filter_module
```

> nginx服务脚本

```
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network.target remote-fs.target nss-lookup.target
[Service]
Type=forking
PIDFile=/usr/local/nginx/logs/nginx.pid
ExecStartPre=/usr/local/nginx/sbin/nginx -t -c /etc/nginx/nginx.conf
ExecStart=/usr/local/nginx/sbin/nginx -c /etc/nginx/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true
```

> nginx配置文件语法高亮

```
#下载高亮插件脚本-->>http://www.vim.org/search.php
#创建插件存放目录
mkdir -pv ~/.vim/syntax
#安装插件
cp nginx.vim ~/.vim/syntax
#创建配置文件
vim ~/.vim/filetype.vim
au BufRead,BufNewFile /etc/nginx/*,/usr/local/nginx/conf/* if &ft == '' | setfiletype nginx | endif
```

nginx配置

main段配置

> 指定以什么身份，或组运行worker进程，GROUP_NAME可以省略

```
user USER_NAME GROUP_NAME;
```

> nginx运行的pid文件，如果编译时不指定，默认在logs目录内

```
pid /path/to/pid.file;
```

> 所有工作进程的最大打开文件数的限制

```
worker_rlimit_nofile 65535;
```

> 所有工作进程的核心文件的最大大小的限制，通常不需要调整

```
worker_rlimit_core SIZE;
```

> 定义工作进程数量。为了避免进程上下文切换，消耗资源，可将工作进程数量设置为和CPU核心数一样，或者比CPU核心数少1的数量，如果将值设置为auto，nginx将自动检测设置值

```
worker_processes  N;
```

> 进程和CPU的绑定，目的是使CPU处理固定的进程，避免CPU缓冲频繁刷新，进程切换、造成CPU资源的消耗默认没有绑定到任何CPU。值auto将自动把进程绑定到可用CPU

```
worker_cpu_affinity cpumask ...;
```

> 降低时间解析度

```
timer_resolution 100ms;
```

> 定义工作进程的调度优先级，即nice值：负值number表示较高的优先级。允许范围通常在-20到20之间

```
worker_priority -10;
```

> 错误日志。debug, info, notice, warn, error, crit, alert, emerg.默认级别的error，则error之后级别的信息都将写入日志文件，debug级别需要编译时指定参数--with-debug

```
error_log file [level];
```

event段配置

> 默认是关闭。如果accept_mutex启用，worker工作进程将轮流接受新的连接。否则，将通知所有worker工作进程有新的连接请求，如果连接数很少的场景下，这个过程可能会浪费系统资源

```
accept_mutex on | off;
```

> 如果accept_mutex已启用，则如果一个工作进程正在接受连接时，工作进程将尝试接受另一个新连接的等待最长时间

```
accept_mutex_delay 500ms;
```

> 指定nginx处理连接的使用的方法，通常不需要明确指定它，nginx默认会使用最有效的方法。method例如：select、poll、epoll等

```
use method;
```

> 单个工作进程可以同时处理的最大连接数(main段),最大连接数是worker_connections x worker_processes的乘积的数量，此值默认是512

```
worker_connections number;
```

http段配置

> 虚拟主机

```
server {
    root /var/www;
    listen 80;
    sever_name www.xuejinwei.com;
}
```

> 设置资源路径映射，即请求URL所对应的资源路径。可以在http, server, location, if in location段定义

```
root /path/to/file;
```

> 可以用在server、location段。根据用户所访问的URL匹配并进行控制处理。location块可以出现多次。
>
> 优先级从大到小依次：=：精确匹配|^~：URL的前半部分匹配|~：正则表达式匹配，区分大小写|~*：正则表达式匹配，不区分大小写| ：不带任何符号|@:定义location区段，这些区段客户端不能访问，只可以由内部产生的请求来访问

```
location [ = | ~ | ~* | ^~ ] URL {
    ...
}
#官方示例
location = / {
    [ configuration A ]
}
location / {
    [ configuration B ]
}
location /documents/ {
    [ configuration C ]
}
location ^~ /images/ {
    [ configuration D ]
}
location ~* \.(gif|jpg|jpeg)$ {
    [ configuration E ]
}
#The “/” request will match configuration A, the “/index.html” request will match configuration B,
#the “/documents/document.html” request will match configuration C,
#the “/images/1.gif” request will match configuration D,
#and the “/documents/1.jpg” request will match configuration E.
```

> 路径别名，只能定义在location段。定义路径别名。root定义所在location所对应的根目录

```
location /images/ {
    root /www/host1;
}   # 表示在/www/host1/ + images/目录下查找所请求的资源
location /images/ {
    alias /www/alias;
}   # 表示在/www/alias/目录下查找所请求的资源
```

> 默认首页文件。可配置于http、server、location段。以顺序查找

```
index file ...;
```

> 错误页面。例如当错误状态码的404，如果定义了404=200，则返回给用户的状态码的200

```
error_page status_code[ = status_code ] ...  URL;
```

> 访问控制 deny、allow 在 location 段定义 可以定义白名单和黑名单

```
allow IP/NETMASK;
deny all|IP;
```

> 基于用户认证

```
Auth_basic "提示信息";
Auth_basic_user_file
"/usr/local/nginx/conf/userfile";
#用户必须还是服务器本地用户：
vim /usr/local/nginx/conf/userfile
xuekaixin:$apr1$YQ3xqzpk$Rf7QIOpuHtLnOzSJpe8Je1
```

> SSL配置

```
server {
    listen       443 ssl;            # 监听端口
```

```
    server_name  localhost;          # 主机名

    ssl_certificate      cert.pem;  # 证书路径
    ssl_certificate_key  cert.key;  # 私钥文件路径

    ssl_session_cache    shared:SSL:1m; # 使用共享内存
    ssl_session_timeout  5m;         # 会话超时时间
    ssl_ciphers  HIGH:!aNULL:!MD5;  # 配置加密套件，定义算法
    ssl_prefer_server_ciphers  on;  # 有限采取服务器算法
    location / {
        root   html;
        index  index.html index.htm;
    }
}
#使用全站加密，http自动跳转https
rewrite ^(.*) https://$host$1 permanent;
```

> 指定包含的配置文件，不同功能的配置文件分离，便于管理与配置

```
include /path/to/*.conf
```

> nginx状态监控，仅能用于location段配置

```
location /status {
    stub_status on;
    allow 192.168.1.100/24;
    deny all;
}
#页面结果
Active connections: 1                    # 活动连接数，当前所有处于打开的状态连接数
server accepts handled requests          # 已经接受的连接;已经处理的连接;已经处理的请求;
 17 17 84
Reading: 0 Writing: 1 Waiting: 0         #处于接受状态的连接;正处于发送响应或处理接受请求;保持连接状态数
```

> rewrite，URL重写。
>
> 语法格式：`rewrite regex(正则表达式) replacement(替换) flag(标志);`
>
> 正则表达式：
>
> `^`必须以^后的实体开头
> `$` 必须以$前的实体结尾
> `.`匹配任意字符
> `[]` 匹配[]内的字集内的任意字符
> `[^]`匹配不包括[]内的字集内的任意字符
> `|`匹配|之前或之后的实体
> `()` 组成一组用于匹配的的实体
>
> flag：
>
> last：匹配后不在对其他规则匹配，改写URL后重新以新的URL请求，并对新的URL再次进行匹配检查
> break：一旦RUL重写完成后，终止rewrite，不在继续匹配
> redirect：返回临时重定向的http状态302
> permanent：放回永久重定向的http状态301

```
#示例
```

```
#将访问/images目录下的图片跳转到/imges目录下
rewrite ^/images/(.*\.jpg)$ /imgs/$1 break;
#跨站跳转
rewrite ^/images/(.*\.php)$
http://www.xuejinwei.me/$1 redirect;
#基于浏览器分离跳转
if($http_user_agent ~ Firefox){
        rewrite ^(.*$) /firefox/$1 break;
}
if($http_user_agent ~ MSIE){
        rewrite ^(.*$) /MSIE/$1 break;
}
if($http_user_agent ~ Chrome){
        rewrite ^(.*$) /chrome/$1 break;
}
```

> 网络连接相关配置

```
keepalive_time Ns;          # 长连接超时时长
keepalive_requests  Ns;     # 一个长连接上所能请求的最大连接数
keepalive_diable NAME;      # 为指定浏览器禁用长连接
tcp_nodelay on|off;         # 是否对长连接使用tcp_nodelay功能
client_header_timeout Ns;   # 读取请求报文的首部的超时时长
client_body_timeout Ns;     # 读取请求报文是实体部分的超时时长
send_timeout Ns;            # 发送响应报文的超时时长
```

> nginx与fastcgi结合

```
location ~ \.php$ {
    root           html;
    fastcgi_pass   127.0.0.1:9000;
    fastcgi_index  index.php;
    fastcgi_param  SCRIPT_FILENAME
 /scripts$fastcgi_script_name;
    include        fastcgi_params;
}
```

参考

nginx官方文档：http://nginx.org/en/docs/