

# GNU/Linux-Nginx



## Nginx 配置与应用

# GNU/Linux-Nginx

## Nginx

Nginx 是一个 HTTP 服务器



# GNU/Linux-Nginx

## Nginx 特点

1. 作为 WEB 服务器，处理静态文件、索引文件，自动索引的效率非常高
2. 作为代理服务器，可实现无缓存的反向代理加速，提高网站运行速度
3. 作为负载均衡服务器，即可以在内部直接支持 Rails( 一种 WEB 开发 ) 和 PHP ，也可以支持代理服务器对外提供服务，同时还支持简单的容错和使用算法完成负载均衡

# GNU/Linux-Nginx

## Nginx 特点

4. 性能方面：效率突出，使用 poll/epoll 模型（一种轮询方法）可支持更高的并发连接。最大可以支持对 50,000 个并发连接数的响应。并且占用内存相当低。

5. 稳定方面：采取分阶段分配技术，使得 CPU 与内存的占用率非常低。比如当有 10,000 个没有活动的链接仅仅占用 2.5M 内存。这样使得像 DoS 的攻击基本没有任何作用。

# GNU/Linux-Nginx

## Nginx 特点

6. 高可用方面：支持热部署，启动速度极快，可以在不间断服务下提供升级。

# GNU/Linux-Nginx

## Nginx 与 Apache 对比

相同点：

1. 都属于 HTTP 服务
2. 都采用模块化结构
3. 都支持通用的语言接口



# GNU/Linux-Nginx

## Nginx 与 Apache 对比

相同点：

4. 都支持正向代理及反向代理

5. 都支持虚拟主机、URL 重写、压缩传输、SSL 加密等



# GNU/Linux-Nginx

## Nginx 与 Apache 对比

不同点：

1. Apache 模块是动态的 , Nginx 模块属于静态
2. Apache 对 FastCGI( 动态脚本如 PHP) 支持不如 Nginx
3. Nginx 支持 epoll 模式，允许高并发量而 Apache 不支持



# GNU/Linux-Nginx

## Nginx 与 Apache 对比

不同点：

4.Nginx 软件包小于 Apache 软件包

5.Nginx 配置文件精短、默认功能少 ,Apache 配置文件相对复杂、清晰。

6.Nginx 占用内存资源极少



# GNU/Linux-Nginx

## Nginx 与 Apache 对比

不同点：

7. 抗并发 :Nginx 处理请求是异步非阻塞方式，而 apache 则是阻塞型的，在高并发下 nginx 能保持低资源低消耗高性能。高度模块化的设计，编写模块相对简单， Nginx 社区活跃，各种高性能模块出品迅速

# GNU/Linux-Nginx

## Nginx 与 Apache 对比

不同点：

8.rewrite:Apache 比 nginx 的 rewrite 强大的

9. 动态页面 :apche 模块超多，基本想到的都可以找到 bug 少，超稳定。 Nginx 的 bug 相对较多。

# GNU/Linux-Nginx

## Nginx 与 Apache 对比

总结：

1. 需要性能的 web 服务，用 nginx 。如果不需要性能只求稳定，那就 apache
2. 作为 Web 服务器：相比 Apache ， Nginx 使用更少的资源，支持更多的并发连接，体现更高的效率。

# GNU/Linux-Nginx

## Nginx 与 Apache 对比

总结：

3. Nginx 配置简洁，Apache 复杂。Nginx 静态处理性能比 Apache 高 3 倍以上。Apache 对 PHP 支持比较简单，Nginx 需要配合其他后端用 Apache 的组件比 Nginx 多

# GNU/Linux-Nginx

## Nginx 与 Apache 对比

总结：

4. 最核心的区别在于 apache 是同步多进程模型，一个连接对应一个进程，nginx 是异步的，多个连接（万级别）可以对应一个进程

5. nginx 处理静态文件好，耗费内存少



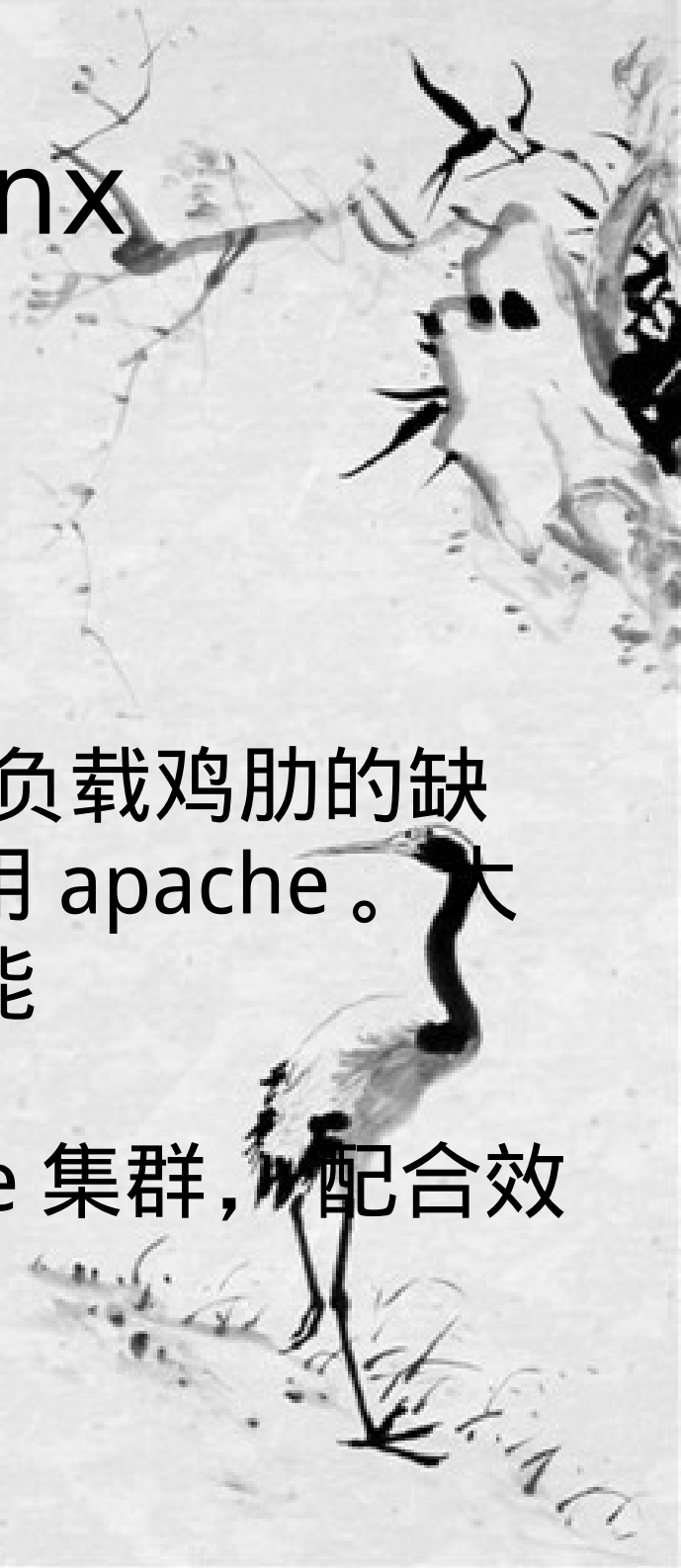
# GNU/Linux-Nginx

## Nginx 与 Apache 对比

总结：

6.apache 有先天不支持多核心处理负载鸡肋的缺点，建议使用 nginx 做前端，后端用 apache。大型网站建议用 nginx 自代的集群功能

7. 前端 nginx 抗并发，后端 apache 集群，配合效果杠杠的。



# GNU/Linux-Nginx

## Nginx 与 Apache 对比

总结：

8.nginx 处理动态请求是不好，动态请求要 apache 去做，nginx 只适合静态和反向。



# GNU/Linux-Nginx

## Nginx 与 Apache 对比

总结：

9.Nginx 优于 apache 的主要两点

1)Nginx 本身就是一个反向代理服务器

2)Nginx 支持 7 层负载均衡

Nginx 可能会比 apache 支持更高的并发，

10. 市场占用率 :Apache 依然占有 62.71% ，而 Nginx 是 7.35% 。 Apache 社区比 Nginx 成熟许多。


# GNU/Linux-Nginx



## Nginx 与 Apache 对比

总结：

11. Apache 在处理动态有优势，Nginx 并发性比较好，CPU 内存占用低，如果 rewrite 频繁，那还是 Apache 吧



# GNU/Linux-Nginx

## Nginx

### 一：安装并启动 Nginx

#### 1. 安装

```
#yum --enablerepo=epel install nginx
```

#### 2. 修改 Nginx 的 FQDN

```
#vi /etc/nginx/nginx.conf
```

```
//*38 行
```

```
server_name www.niliu.edu;
```



# GNU/Linux-Nginx

## Nginx

一：安装并启动 Nginx

3. 启动 Nginx

```
#systemctl start nginx
```

```
#systemctl enable nginx
```

4. 客户端浏览器测试



# GNU/Linux-Nginx

## Nginx

一：安装并启动 Nginx

/\* 如果打算不间断服务的前提下重新启动 Nginx 服务（平滑重启），可使用如下命令

```
#kill -HUP `cat /run/nginx.pid`
```

/\*HUP 信号为重新加载配置，即关闭原有进程并开启新的进程



# GNU/Linux-Nginx

## Nginx

### 二：虚拟主机

#### 1. 编辑 nginx 虚拟主机配置文件

```
#vi /etc/nginx/conf.d/niliu.host.conf
```



# GNU/Linux-Nginx

## Nginx

### 二：虚拟主机 ( 基于 FQDN )

```
Server {  
    /* 定义监听端口  
    listen    80;  
  
    /* 定义虚拟主机的 FQDN  
    server_name www.niliu.edu;
```



# Nginx GNU/Linux-Nginx

## 二：虚拟主机 ( 基于 FQDN )

```
Location / {  
    /* 定义内容根目录  
    root /usr/share/nginx/www;  
  
    /* 定义所以文件  
    index index.html index.htm;  
}  
}
```





# GNU/Linux-Nginx

## Nginx

二：虚拟主机 ( 基于 FQDN)

```
#mkdir -pv /usr/share/nginx/www
```

```
#echo "test" >>
```

```
/usr/share/nginx/www/index.html
```

```
/* 浏览器测试
```



# GNU/Linux-Nginx

## Nginx

三 :https

/\* 设定私钥

```
#cd /etc/pki/tls/certs
```

```
#make server.key
```

/\* 将私钥密码脱离

```
#openssl rsa -in server.key -out server.key
```



# GNU/Linux-Nginx

## Nginx

三 :https

/\* 生成并签署证书

#make server.csr

#openssl x509 -in server.csr -out server.crt -req  
-signkey server.key -days 3650



# GNU/Linux-Nginx

## Nginx

三 :https

```
/* 编辑 Nginx  
#vi /etc/nginx/nginx.conf
```

```
/* 在 server 区段中加入以下内容
```



# GNU/Linux-Nginx

## Nginx

```
≡ :https
server {
    listen      80 default_server;
    listen     [::]:80 default_server;
```

/\* 指定 https 监听端口

```
listen      443 ssl;
```



# GNU/Linux-Nginx

## Nginx

≡ :https

```
server_name www.server.world;  
root      /usr/share/nginx/html;  
/* 添加证书及私钥  
ssl_certificate      /etc/pki/tls/certs/server.crt;  
ssl_certificate_key  
/etc/pki/tls/certs/server.key;
```

# GNU/Linux-Nginx

## Nginx

≡ :https

#systemctl restart nginx

浏览器测试

[https://nginx\\_server\\_ip](https://nginx_server_ip)



# GNU/Linux-Nginx

## Nginx

### 四：认证

```
# yum -y install httpd-tools
```

```
# vi /etc/nginx/nginx.conf  
/* 在 server 区段中添加
```





# GNU/Linux-Nginx

## Nginx

### 四：认证

```
location /auth-basic {  
    auth_basic          "Basic Auth";  
    auth_basic_user_file  
"/etc/nginx/.htpasswd";  
}
```



# GNU/Linux-Nginx

## Nginx

### 四：认证

```
#htpasswd -c /etc/nginx/.htpasswd snow
```

```
#systemctl restart nginx
```

### 浏览器测试

```
http://nginx_srv_ip/auth-basic
```



# GNU/Linux-Nginx

## Nginx

五：支持 PHP/PHP-FPM(FastCGI)

FastCGI: 属于一个可伸缩、高速的 HTTP server 和动态脚本语言间之间的通信接口

FastCGI 是从 CGI 发展改进而来的。传统的 CGI 速度慢，性能差，无法支持高并发这些问题在 FastCGI 上都得到了完美的解决。



# GNU/Linux-Nginx

## Nginx

### 五：支持 PHP/PHP-FPM(FastCGI)

FastCGI 属于 C/S 结构。这意味着无需将脚本解析服务器与 http 服务器放在一起（像 apache 与 tomcat）

如果 http 服务器遇到动态程序时就直接递交给 FastCGI 进程来执行，将得到的结构返回客户端。


# GNU/Linux-Nginx



## Nginx

### 五：支持 PHP/PHP-FPM(FastCGI)

PHP-FPM 属于 PHP 的 FastCGI 的进程管理程序，属于 PHP 的一个补丁。这样在提升处理性能方面更加优秀。



# GNU/Linux-Nginx



## Nginx

### 五：支持 PHP/PHP-FPM(FastCGI)

在部署上，可以将 Nginx 作为静态或动态请求的转发，而部署在不同的服务器上 PHP/PHP-FPM 则专门解析 PHP 解析。

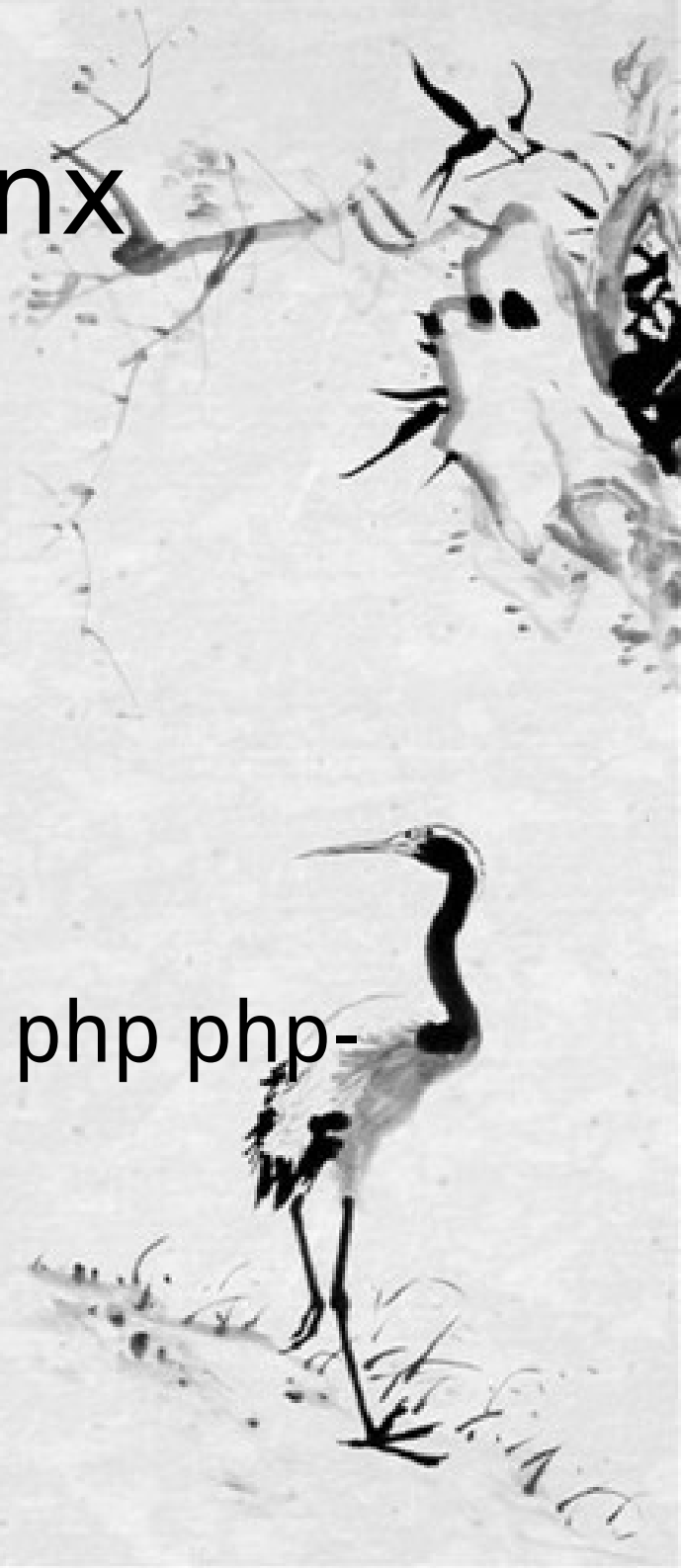
# GNU/Linux-Nginx

## Nginx

五：支持 PHP/PHP-FPM

/\* 安装 PHP 及 PHP-FPM

```
# yum --enablerepo=epel -y install php php-  
mbstring php-pear php-fpm
```



# GNU/Linux-Nginx

## Nginx

### 五：配置 PHP-FPM

```
/* 安装 PHP 及 PHP-FPM  
# vi /etc/php-fpm.d/www.conf  
/* 修改 39 行  
user = nginx  
  
/* 修改 41 行  
group = nginx
```





# GNU/Linux-Nginx

## Nginx

### 五：配置 PHP-FPM

```
#systemctl start php-fpm  
#systemctl enable php-fpm
```



# GNU/Linux-Nginx

## Nginx

### 五：配置 PHP-FPM

```
#vi /etc/nginx/nginx.conf
```

/\* 在 server 区段增加如下内容

```
location ~ \.php$ {  
    fastcgi_pass 127.0.0.1:9000;  
    fastcgi_param SCRIPT_FILENAME  
$document_root$fastcgi_script_name;
```



# GNU/Linux-Nginx

## Nginx

### 五：配置 PHP-FPM

```
fastcgi_param  PATH_INFO $fastcgi_path_info;  
    include     fastcgi_params;  
}
```

```
#systemctl restart nginx
```



# GNU/Linux-Nginx

## Nginx

### 五：配置 PHP-FPM

```
#echo "<?php phpinfo() ?>" >  
/usr/share/nginx/html/test.php
```

### 浏览器测试

[https://nginx\\_srv\\_ip/test.php](https://nginx_srv_ip/test.php)



# GNU/Linux-Nginx



## Nginx

六：反向代理 (nginx<->apache)

反向代理：位于 Internet 的客户端通过企业边界设备（路由）获取企业内部的 WEB 服务器的资源。



# GNU/Linux-Nginx

## Nginx

六：反向代理 (nginx<->apache)

1) 安装一台 nginx 做反向代理服务

2) 安装第二台 apache 做后台 www 服务

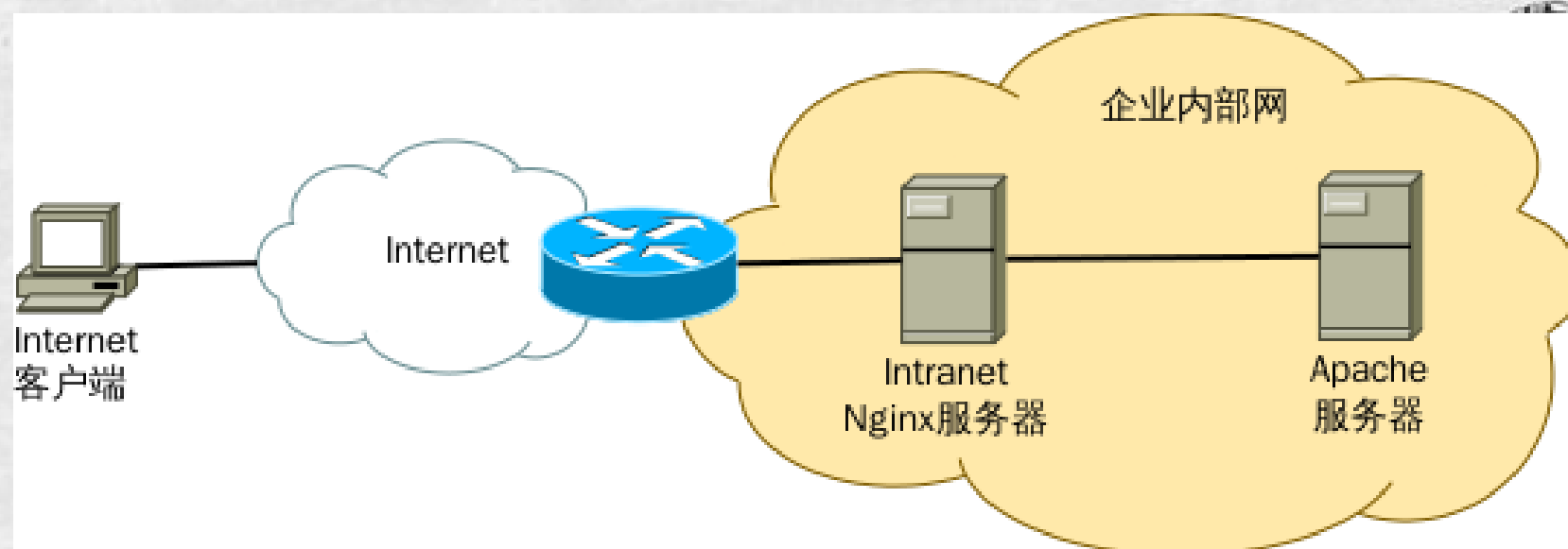


# GNU/Linux-Nginx

## Nginx

### 六：反向代理 (nginx<->apache)

#### 示意图



# GNU/Linux-Nginx

## Nginx

六：反向代理 (nginx<->apache)

3) 编辑 nginx 反向代理服务器的配置文件  
#vi /etc/nginx/nginx.conf





# GNU/Linux-Nginx

## Nginx

六：反向代理 (nginx<->apache)

3) 编辑 nginx 反向代理服务器的配置文件  
#vi /etc/nginx/nginx.conf

//\* 在 server 区段增加如下内容



# GNU/Linux-Nginx

## Nginx

六：反向代理 (nginx<->apache)

```
server {  
  
    listen    80 default_server;  
    listen    [::]:80 default_server;  
    server_name test.niliu.edu;
```



# GNU/Linux-Nginx

## Nginx

六：反向代理 (nginx<->apache)

```
proxy_redirect      off;
proxy_set_header    X-Real-IP
$remote_addr;
proxy_set_header    X-Forwarded-For
$proxy_add_x_forwarded_for;
proxy_set_header    Host $http_host;
```

# GNU/Linux-Nginx

## Nginx

六：反向代理 (nginx<->apache)

```
location / {  
    proxy_pass http://test1.niliu.edu/;  
}  
}
```

```
# systemctl restart nginx
```



# Nginx GNU/Linux-Nginx

六：反向代理 (nginx<->apache)

/\* 修改 apache 服务

#vi /etc/httpd/conf/httpd.conf

/\* 修改 196 行如下

LogFormat "%X-Forwarded-For" %i

%l %u %t "%r" %>s %b "%{Referer}i" \"%  
{User-Agent}i\" combined

#systemctl restart httpd



# GNU/Linux-Nginx

## Nginx

六：反向代理 (nginx<->apache)

浏览器测试

<http://test.niliu.edu>

应看到

test1.niliu.edu 主机的内容



# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

1)nginx 反向代理服务器 1 台

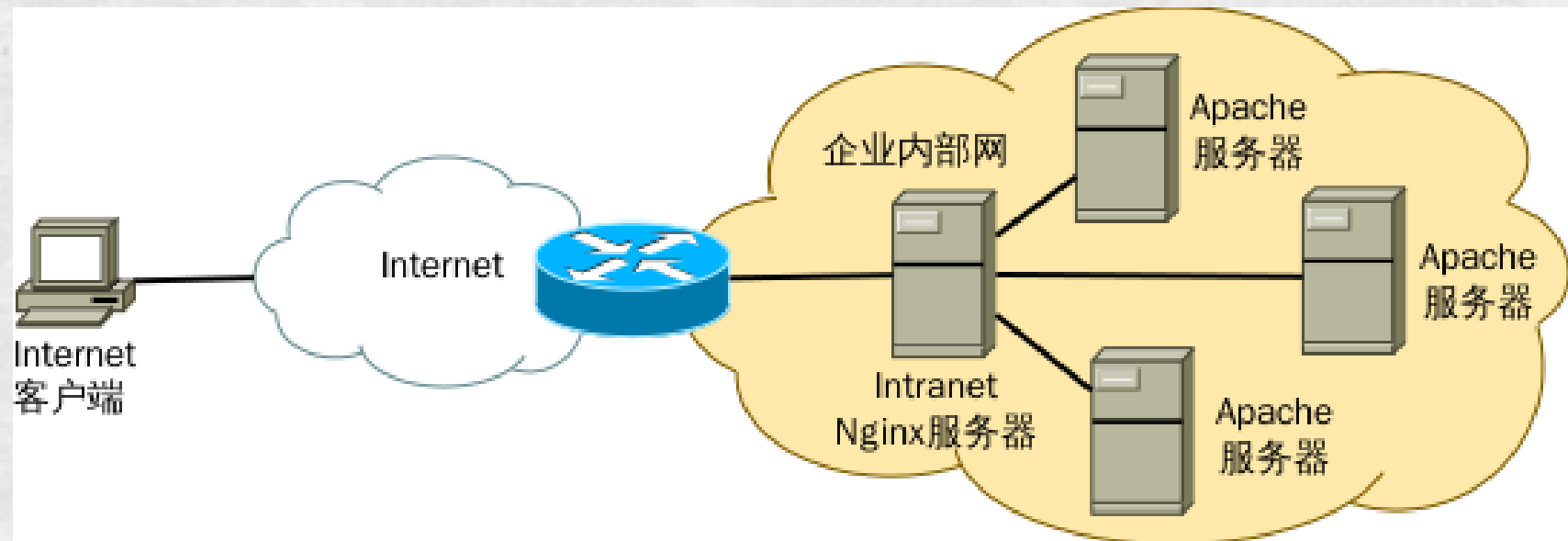
2) 三台 apache 服务并运行，每台页面内容不一样  
以便于区分

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

示意图





# GNU/Linux-Nginx



## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

3) 配置 nginx

```
# vi /etc/nginx/nginx.conf
```

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

3) 配置 nginx

/\* 于 http 区段，设定负载均衡组



# GNU/Linux-Nginx

## Nginx

```
http {  
    upstream backends {  
        server test1.niliu.edu:80 weight=3 max_fails=3  
fail_timeout=20s;  
        server test2.niliu.edu:80 weight=2 max_fails=3  
fail_timeout=20s;  
        server test3.niliu.edu:80 weight=1 max_fails=3  
fail_timeout=20s;  
    }
```

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

4) 配置 nginx

/\* 设置反向代理至负载均衡组



# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

3) 配置 nginx

```
server {  
    listen    80 default_server;  
    listen    [::]:80 default_server;  
    server_name test.niliu.edu;
```



# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

3) 配置 nginx

```
proxy_redirect      off;
proxy_set_header    X-Real-IP
$remote_addr;
proxy_set_header    X-Forwarded-For
$proxy_add_x_forwarded_for;
proxy_set_header    Host $http_host;
```

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

3) 配置 nginx

```
location / {  
    proxy_pass http://backends;  
}  
}
```

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

### 4) 测试

```
#systemctl restart nginx
```

浏览器测试：

<http://test1.niliu.edu> ← 多执行几次应显示出后面  
各 apache 的内容



# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

/\* 在测试时，可当掉一个 httpd 时,nginx 会自动得知并不在转到断开的 httpd 上，当修复好后台损坏的 httpd 时，等待一会 nginx 可自动得知修复好的 httpd 上线，测试会自动连接 \*/

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

/\* 也可添加如下字段让 nginx 做负载故障迁移

```
location / {  
    proxy_pass http://backends;  
    proxy_next_upstream http_500 http_502  
http_503 error timeout invalid_header;  
    include /etc/nginx/proxy.conf  
}
```

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

```
/*proxy_next_upstream http_500 http_502  
http_503 error timeout invalid_header;
```

/\* 表示：当 nginx 发现后端服务节点返回 500/502/503 错误时，自动将请求转发到负载均衡集群组中的另一个服务器上，实现故障转移

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

```
/*include /etc/nginx/proxy.conf
```

```
/* 设定反向代理配置，以优化代理能力
```



# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

// 在 proxy.conf 中可由一下内容：

proxy\_redirect off; ← 是否对发送给客户端的 URL 进行修改

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

// 如 Location: http://test.abc.com:9080/abc.html  
。因为 nginx 服务器侦听的是 80 端口，所以这样的 URL 给了客户端，必然会出错。针对这种情况，加一条 proxy\_redirect 指令：proxy\_redirect http://test.abc.com:9080/ /，把所有“http://test.abc.com:9080/”的内容替换成“/”再发给客户端

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

/\* 设置由后端服务器获取用户的主机名或真实 IP  
及代理的 IP

```
proxy_set_header Host $host;
```

```
proxy_set_header X-Real-IP $remote_addr
```

```
proxy_set_header X-Forwarded-For  
$proxy_add_x_forwarded_for;
```



# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

/\* 用于指定客户端请求主体缓冲区大小，可理解为先保存到本地在传送给用户

client\_body\_buffer\_size 128k;

/\* 表示与后端服务器连接超时时间，从发起握手开始，单位为秒

proxy\_connect\_timeoute 90



# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

/\* 后端服务器传回数据的超时时间，如超过则断开连接

```
proxy_send_timeout 90;
```

/\*nginx 等待后端服务器处理的等候超时时间

```
proxy_read_timeoute 90;
```

# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

/\* 设置缓冲区大小 ( 等于 proxy\_buffers 设置的大小 )

proxy\_buffer\_size 4k;

/\* 设置缓冲区的数量及大小

proxy\_buffers 4 32k;



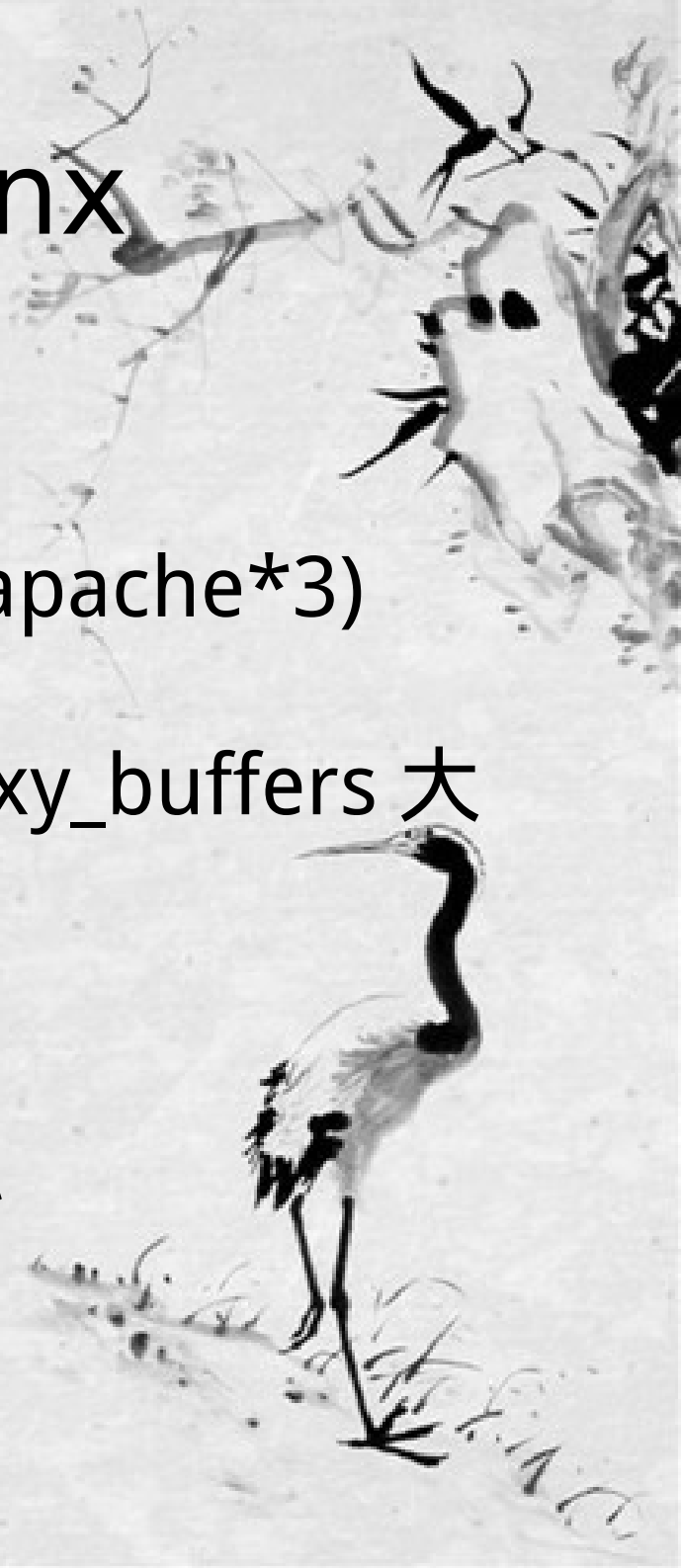
# GNU/Linux-Nginx

## Nginx

七：反向代理及负载均衡 (nginx<->apache\*3)

```
/* 设置系统繁忙时可以使用的 proxy_buffers 大小 . 官方推荐为 proxy_buffersx2  
proxy_busy_buffers_size 64k;
```

```
/* 指定 proxy 缓存临时文件的大小  
proxy_temp_file_write_size 64k;
```



# GNU/Linux-Nginx

## Nginx

### 八：防盗链功能

```
#vi /etc/nginx/nginx.conf  
...
```

```
/* 指定防盗链的后缀名
```

```
Location ~* \.(exe|zip|rar|7z)$ {
```

```
    /* 指定 niliu.net 可以方位这些资源  
    valid_referers none blocked *.niliu.net  
    niliu.net;
```



# Nginx GNU/Linux-Nginx

## 八：防盗链功能

/\* 设定无效链接

```
If ($invalid_referer) {
```

```
/* 如果属于无效链接则转至指定页面
```

```
Rewrite ^/ http://www.niliu.net/err/error.html;
```

```
/* 如果属于无效链接则返回 403 错误
```

```
#return 403;
```

```
}
```

```
}
```



# GNU/Linux-Nginx

## Nginx

### 九：日志分割

/\*nginx 没有像 Apache 一样的日志分割功能，所以需要写一个脚本来实现日志的分割

```
#!/bin/bash  
savepath=' /home/snow/logs'  
nglogs=' /var/lig/nginx/'
```



# GNU/Linux-Nginx

## Nginx

### 九：日志分割

```
mkdir -p $savepath/$(date +%Y)/$(date +%m)
mv $nglogs/access.log $savepath/$(date +%Y)/$(date +%m)/access.$(date +%Y%m%d).log
mv $nglogs/error.log $savepath/$(date +%Y)/$(date +%m)/error.$(date +%Y%m%d).log
```

# GNU/Linux-Nginx

## Nginx

九：日志分割

```
kill -USR1 `cat /run/nginx.pid`
```





# GNU/Linux-Nginx



## Nginx

### 九：日志分割

将脚本存放到 cron 中，用于每天的调用。



# GNU/Linux-Nginx

## Nginx

十 :Nginx 内核参数优化

将以下参数写入至 /usr/lib/sysctl.d/00-system.conf 文件中



# GNU/Linux-Nginx

## Nginx

十 :Nginx 内核参数优化

/\* 设定 tcp 链接的 timewait 数量

net.ipv4.tcp\_max\_tw\_buckets = 6000

/\* 设定用来允许系统打开的端口范围

net.ipv4.ip\_local\_port\_range = 1024 65000



# GNU/Linux-Nginx

## Nginx

十 :Nginx 内核参数优化

/\* 是否启用 timewait 快速回收机制 (1 为允许)  
net.ipv4.tcp\_tw\_recycle = 1

/\* 是否允许将 TIME-WAIT sockets 重新用于新的  
TCP 链接  
net.ipv4.tcp\_tw\_reuse = 1



# GNU/Linux-Nginx

## Nginx

十 :Nginx 内核参数优化

```
/* 当 SYN 等待队列溢出时是否启用 Cookies 处理  
net.ipv4.tcp_syncookies = 1
```

```
/* 设订 TCP 并发连接数  
net.core.somaxconn = 262144
```




# GNU/Linux-Nginx



## Nginx

### 十 :Nginx 内核参数优化

/\* 当每个网络接口接收速度比内核处理速度快时，允许发送到队列的数据包的最大数量  
net.core.netdev\_nax\_backlog = 262144



# GNU/Linux-Nginx

## Nginx

### 十 :Nginx 内核参数优化

/\* 设定系统中最多有多少个 TCP Socket 不被关联到任何一个用户文件句柄上 . 如超过则被复位并显示警告信息 . 主要用于防止简单的 DoS 攻击  
net.ipv4.tcp\_max\_orphans = 262144

# GNU/Linux-Nginx

## Nginx

十 :Nginx 内核参数优化

/\* 设定用于记录那些尚未收到客户端确认信息的  
链接请求的最大值

```
net.ipv4.tcp_max_syn_backlog = 262144
```

/\* 设定内核放弃链接之前发送 syn+ack 的数量

```
net.ipv4.tcp_synack_retries = 1
```





# GNU/Linux-Nginx

## Nginx

### 十 :Nginx 内核参数优化

/\* 设定内核放弃链接之前发送 syn 的数量  
net.ipv4.tcp\_syn\_retries = 1

/\* 设定套接字保持在 FIN-WAIT-2 状态的时间。以避免产生大量的死套接字而产生的内存溢出风险  
net.ipv4.tcp\_fin\_timeout = 1

# GNU/Linux-Nginx



## Nginx

### 十 :Nginx 内核参数优化

/\* 当 keepalive 启动时 ,tcp 发送 keepalive 消息的  
频率 .( 单位为小时 )

```
net.ipv4.tcp_keepalive_time = 30
```



# GNU/Linux-Nginx

## Nginx

十一 :Nginx 实时监控工具 --ngxtop

### 1. 源代码安装

```
#wget
```

```
http://pypi.python.org/packages/source/s/setuptools/setuptools-0.6c11.tar.gz
```

```
#tar zxvf setuptools-0.6c11.tar.gz
```

```
#cd setuptools-0.6c11
```

```
#python setup.py build
```

```
#python setup.py install
```



# Nginx

# GNU/Linux-Nginx

十一 :Nginx 实时监控工具 --ngxtop

1. 源代码安装

```
#wget
```

```
https://pypi.python.org/packages/source/p/pip/pip-
```

```
7.1.2.tar.gz#md5=3823d2343d9f3aaab21cf9c917710196
```

```
#tar zxvf pip-7.1.2.tar.gz
```

```
#cd pip-7.1.2
```

```
#python setup.py install
```

# GNU/Linux-Nginx

## Nginx

十一 :Nginx 实时监控工具— ngxtop

### 1. 源代码安装

```
#pip install ngxtop
```

```
#ngxtop
```



# GNU/Linux-Nginx

## Nginx

十一 :Nginx 实时监控工具— ngxtop

2.ngxtop 语法

ngxtop [options]

ngxtop [options] (print | top | avg | sum) <var>

ngxtop info



# Nginx GNU/Linux-Nginx

十一 :Nginx 实时监控工具— ngxtop

## 3.ngxtop 参数

- l: 指定日志文件的完整路径 (Nginx 或 Apache2)
- f: 日志格式
- no-follow: 处理当前已经写入的日志文件，而不是实时处理新添加到日志文件的日志
- t: 更新频率

# Nginx GNU/Linux-Nginx

## 十一 :Nginx 实时监控工具— ngxtop

### 3.ngxtop 参数

- n : 显示行号
- o : 排序规则 ( 默认是访问计数 )
- a ..., -a ... : 添加表达式 ( 一般是聚合表达式  
如: sum, avg, min, max 等 ) 到输出中。
- v: 输出详细信息
- i: 只处理符合规则的记录

