路由器：具有路由表（记录下一跳的IP地址、网关、子网掩码等）和ARP（Address Resolution Protocol地址解析协议）表（记录下一跳的IP地址和物理Mac地址等）可以连接不同网络段，可以转发不同网络段的信息，由于转发不同网络段的消息，下一跳的Mac物理地址会变化，转发下一跳信息的时候会忽略Mac地址

交换机：二层设备没有路由表只有类似ARP表的数据表，不需要配置IP地址，只负责信息的转发（转发依据是计算机的网卡Mac物理地址），只能连接同一网络段的终端机器，转发下一跳信息时会根据逻辑IP地址和物理Mac地址到达指定主机

终端计算机的Gateway是网关或者是路由表，记录终端计算机的下一跳即计算机所有信息转发的下一个地址

网关是路由器和交换机的入口和出口，入口和出口是不同的网关IP地址

OSI（Open System Interconnection）开放式系统互联，OSI七层协议模型从上往下：应用层、表示层、会话层、传输层（之前4层为高层定义应用程序的功能，传输层指定传输端口）、网络层（之后3层数据流，网络层指定网络逻辑地址即IP，数据包不传递子网掩码，每个下一跳主机自己的路由表中保存着对应的子网掩码）、数据链路层（网卡硬件地址）、物理层

TCP协议会先做三次握手测试连接，连接后传输数据进行四次分手断开连接

CDN（Content Delivery Network）内容分发网络即静态代理服务器，缓存网络网站不变的内容以节约带宽网络资源和负载均衡

LVS（Linux Virtual Server）Linux虚拟服务器：基于OSI七层协议模型的最底4层即传输层、网络层、数据链路层、物理层进行数据交换，数据交换不进行三次握手连接四次分手断开操作，速度快性能强，能实现访问量的负载均衡不能实现访问内容的负载均衡分配，LVS一般包含分发服务器和服务服务器，分发服务器与服务器之间使用交换机连接，分发服务器与服务服务器都有独立私网IP但公网服务IP一般都是一样的，客户端访问的是公网服务IP，对于客户端整个LVS集群都是同一个公网服务IP地址

分发服务器：一般为了避免单节点挂掉问题分配多个分发服务器有主从等级之分，具有公网服务IP和独立私网IP，功能单一只负责转发数据包，只分析客户端访问IP地址的最底4层协议，不建立三次握手四次分手性能强速度快，记录客户端IP地址端口与服务服务器IP地址端口联系与状态，使用交换机特性的物理地址与服务服务器进行通信

服务服务器：具有公网IP和独立私网IP，负责提供服务，返回网络请求时隐藏服务服务器自己的独立私网IP地址，将IP数据包的IP替换成集群的公网服务器IP返回给客户端

Nginx：基于OSI七层协议模型进行数据交换，数据交换进行三次握手四次分手操作，速度和性能相对低于LVS，能实现访问内容的负载均衡分配

LVS开源技术已经嵌入到Linux的内核中名为ipvs，对应的管理应用程序为ipvsadm

负载均衡器：

硬件：F5 BIG-IP、Citrix NetScaler、A10

软件：基于OSI七层协议模型的底四层（LVS已经嵌入Linux系统内核中）、基于OSI七层协议模型（Nginx、haproxy、httpd）

LVS三种类型：NAT地址转换（S/D 源/目标地址转换，请求相应都会经过分发服务器）、DR（直接路由，请求经过分发服务器，响应不讲过分发服务器）、TUN隧道（在OSI七层协议模型的数据包之上再加入网络层、数据链路层一般用于服务服务器不在同一局域网内，请求经过分发服务器，响应不讲过分发服务器）

VIP：虚拟主机IP

DIP：kernel parameter（内核参数）：

arp\_ignore：定义接受到ARP请求时的响应级别（0/1 本地配置有相应地址给予响应/请求的目标Mac地址配置请求到接口上给予响应）

arp\_announce：定义自己地址向外通告时的通告级别（0/1/2 本地任何接口上的任何地址向外通告/目标网络通告与其网络匹配的地址/本地接口上地址匹配的网络进行通告）

DR模式步骤（ipvs已经加入到Linux内核服务中，不需要手动启动服务，配置完成后自动执行配置的ipvs服务）：

DR分发服务器Linux系统使用命令yum install ipvsadm安装ipvsadm

DR设置两个IP地址：

DIP：设置独立私网静态的IP

VIP：ifconfig 网卡名:网卡号 （虚拟IP地址/数值）/（虚拟IP地址 netmask 子网掩码） 设置DR服务器的虚拟公网服务IP地址（不能与DR分发服务器的独立私网静态IP的子网掩码相同），子网掩码可以通过从左到右的位数为数值指定

多台RS服务服务器设置独立私网静态IP，通过设置Linux内核参数修改报文源IP的设置：

echo 1 > /proc/sys/net/ipv4/conf/网卡名/arp\_ignore

echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/网卡名/ arp\_announce

echo 2 > /proc/sys/net/ipv4/conf/all/ arp\_announce

多台设置RS服务服务器的虚拟IP地址：

ifconfig 网卡名:网卡号 虚拟的IP地址 netmask 子网掩码（不能与RS服务服务器的独立私网静态IP的子网掩码相同） broadcast 广播地址

RS服务服务添加路由：

route add -host IP地址 dev 网卡名:网卡号

DR分发服务器添加路由：

route add -host IP地址 dev 网卡名:网卡号

使用客户端ping连接测试集群虚拟IP地址是否连通

arp -a 查看ARP（Address Resolution Protocol地址解析协议）表

route -n 查看路由表

DR分发服务器使用ipvsadm添加集群服务：

ipvsadm -C 清除ipvs配置

ipvsadm –（S >）/（R <） 文件路径名 保存/加载指定文件路径名的ipvs配置

ipvsadm -A -t 公网服务IP地址:端口 -s wlc 添加监听的公网服务IP地址和端口，指定调度算法（默认wlc算法调度）

ipvsadm -a/e/d（添加/修改/删除） -t/u/f 公网服务IP地址:端口 -r 独立私网IP地址（NAT模式需要指定端口号） [-g/i/m（指定LVS模式DR/TUN/NAT默认DR）] [-w 数值（指定服务器权重）] 为指定公网服务IP地址添加RS服务服务器集群

ipvsadm –[L/l][n/stats/rate/timeout/c] 显示查看ipvs[数字格式显示主机地址和端口/统计数据/速率/显示TCP、TCPFIN和UDP的会话超时时长/当前ipvs连接状况]配置

netstat -natp 查看计算机网络监听端口进程服务

ip addr del IP地址 dev 网卡号 删除指定网卡号上的IP地址

curl IP地址 返回指定IP地址的web文本形式内容

ifconfig 网卡名:网卡号 down 删除指定网卡名网卡号的IP地址

HA（High Available）高可用，通过心跳机制探测后端DR和RS服务器是否可用，不可用从LVS中删除，ipvsadm没有自动删除LVS配置的机制，keepalived通过配置VRRRP提供高可用性具有自动删除LVS配置的机制

keepalived配置：

多台DR服务器安装keepalived，配置文件/etc/keepalived/keepalived.conf，一般编辑配置文件之前复制备份

keepalived.conf文件：

global\_defs {

notification\_email {

email地址 指定服务器出问题后向指定的email地址发送通知

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 192.168.200.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

vrrp\_strict 配置虚拟公网服务IP地址需要删除

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER/BACKUP 指定本机DR服务器的主/后备

interface 网卡名 指定网卡名

virtual\_router\_id 51

priority 数值 指定后备DR服务器的权重，数值越大优先级越高

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

（虚拟公网服务IP地址/数值）/（虚拟公网服务IP地址 netmask 子网掩码） dev 网卡名 label 网卡名:网卡号

}

}

virtual\_server 虚拟公网服务IP地址 端口号 {

delay\_loop 6

lb\_algo rr 指定调度算法

lb\_kind NAT 指定LVS的模式

net\_mask 子网掩码地址 指定子网掩码

persistence\_timeout 50 指定连接保存时长单位秒

protocol TCP

real\_server 独立私网IP地址 端口号 { 指定多台RS服务服务器的心跳机制

weight 1 权重

HTTP\_GET { 心跳机制检测方式

url {

path / 心跳检测的地址

status\_code 200 心跳检测返回的状态

}

connect\_timeout 3 指定超时时间

nb\_get\_retry 3 指定重复检查次数

delay\_before\_retry 3 指定每次检查的时间间隔单位秒

}

}

}

Nginx（engine x）是一个高性能轻量级的Web HTTP和反向代理服务器，也是一个电子邮件IMAP/POP3/SMTP代理服务器，2004年10月4日发布第一个公开版本0.1.0，源代码以类BSD许可证BSD-like协议的形式发布，由俄罗斯程序设计师lgor Sysoev开发，特性有稳定性、丰富的功能集、示例配置文件和低系统资源消耗，官方测试在消耗计算机资源非常低并且运行非常稳定的情况下，Nginx能够支撑5万的并发链接量，使用Nginx服务器做网站的中国大陆有新浪、网易、搜狐、腾讯等

Nginx和Apache的优缺点：

Nginx是轻量级，消耗资源少，抗高并发，异步非阻塞处理请求，设计高度模块化（与Linux内核类似可以定制化需求，可以选择相应模块进行自定义），社区活跃，配置简洁，异步模型，多个连接（万级别）可以对应一个进程

Apache Tomcat是一个重量级容器级服务器，阻塞型处理请求，高并发支持性不好，rewrite比Nginx强，模块多，bug相对比Nginx较少，配置复杂，同步多进程模型，一个连接对应一个进程

正向代理服务器：客户端请求到达不了服务器，正向代理服务器代理客户端请求信息获取请求返回的信息，然后将信息返回给客户端

反向代理服务器：客户端请求可以到达服务器，但请求的信息不在服务器上，服务器需要反向代理去其他的服务器上获取请求需要返回的信息，然后将反向代理获得的请求返回信息返回给客户端

Tengine是Nginx的封装版，由淘宝开发支持开源，Tengine兼容Nginx，官方地址<http://tengine.taobao.org/>

Tengine特性：

动态模块加载（DSO）支持，加入模块不需要重新编译这个Tengine

支持SO\_REUSEPORT选项，建连性能提升为官方Nginx的的三倍

支持SPDY\_v3协议，自动检测同一端口的SPDY请求和HTTP请求

流式上传到HTTP后端服务器或FastCGI服务器，大量减少机器的I/O压力

更加强大的负载均衡能力，包括一致性hash模块、会话保持模块、后端服务器主动健康检查，根据服务器状态自动上线下线，已经动态解析upstream中的域名

输入过滤器机制支持，使Web应用防火墙的编写更方便

支持设置proxy、memcached、fastcgi、scgi、uwsgi在后端失败时的重试次数

动态脚本语言Lua支持，扩展功能非常高效简单

支持管道（pipe）和syslog（本地和远端）形式的日志以及日志抽样

支持按指定关键字（域名，url等）收集Tengine运行状态

组合多个CSS、JavaScript文件的访问请求变成一个请求

自动去除空白字符和注释，从而减少页面的体积

由于Nginx与Linux内核一样可以定制化自定义选择功能，所以一般安装Nginx是使用源码进行编译然后安装：

yum install -y gcc gcc-c++ openssl-devel pcre-devel zlib-devel 安装编译Nginx需要的依赖

groupadd -r nginx 创建Nginx的用户组，为了方便Nginx运行不影响Linux安全

useradd -r -g nginx -M nginx 创建Nginx的用户不创建用户的家目录

./configure \ 检查编译环境是否具备，生成Makefile编译的文档（选项可以通过./configure --help 命令查看）

--prefix=文件路径名 \ 指定文件路径名为编译安装的路径

--sbin-path=文件路径名 \ 指定文件路径名为编译后的启动文件路径

--conf-path=文件路径名 \ 指定文件路径名为编译后的配置文件路径

--error-log-path=文件路径名 \ 指定文件路径名为编译的错误日志文件路径

--http-log-path=文件路径名 \

--pid-path=文件路径名 \

--lock-path=文件路径名 \

--user=用户名 \ 指定Nginx的用户

--group=用户组名 \ 指定Nginx的用户组

--with-http\_ssl\_module \ Nginx加入指定的模块

--with-http\_flv\_module \ Nginx加入指定的模块

--with-http\_stub\_status\_module \ Nginx加入指定的模块

--with-http\_ssl\_module \ Nginx加入指定的模块

--with-http\_gzip\_static\_module \ Nginx加入指定的模块

--http-client-body-temp-path=文件路径名 \

--http-proxy-temp-path=文件路径名 \

--http-fastcgi-temp-path=文件路径名 \

--http-uwsgi-temp-path=文件路径名 \

--http-scgi-temp-path=文件路径名 \

--with-pcre

make && make install 编译并进行安装

为了方便可以在/etc/init.d目录下创建Nginx脚本并赋予执行权限将Nginx注册为Linux的服务，使用chkcofig nginx on命令将Nginx服务设置为开机自启动

Nginx架构是Master/Worker架构，一个Master进程带多个Worker进程，Master进程在Nginx整个生命周期对配置文件的处理和Worker的管理，多个Worker进程（一般设置与CPU核心数一致）处理客户端请求

Apache HTTP每一个请求启动一个Linux系统的进程进行处理，通过Linux系统的内核对进程进行调度，请求通过网卡到达Linux内核空间，通过OSI七层协议模型转换到上层用户空间，上层用户空间的请求处理程序业务逻辑再通过内核空间请求访问硬盘，访问的数据内容通过内核空间加载到内存再放到用户空间内存一份数据内容（Linux内核空间与用户空间隔离不共享），用户空间处理程序从内存数据内容再通过内核空间访问网卡发送数据内容，消耗资源大，Nginx根据CPU核心数启动相同数量的Worker，每个请求分到空闲的Worker上，Worker使用直接内存存取DMA (Direct Memory Access)硬件自动控制暂停原请求对CPU的占用进行处理其他新请求，内存存取完成后发送中断Worker继续处理原请求，Nginx请求通过网卡到达Linux内核空间，通过OSI七层协议模型转换到上层用户空间，上层用户空间的请求处理程序业务逻辑再通过内核空间请求访问硬盘，访问的数据内容直接通过内核空间加载到内存控制网卡发送数据内容

nginx.conf配置文件：

user 用户 用户组; 指定Nginx运行的用户和用户组

worker\_processes 数值/auto; 指定worker进程的数量（一般与CPU核心数一致），auto自动识别系统核心数

#error\_log logs/error.log;（日志路径）

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#error\_log "pipe:rollback logs/error\_log interval=1d baknum=7 maxsize=2G";

#pid logs/nginx.pid; 进程文件路径

events {

[use kqueue/rtsig/epoll/（/dev/poll）/select/poll]; 指定参考事件模型，epoll模型是Linux2.6以上版本内核中的高性能网络I/O模型，运行在FreeBSD上一般使用kqueue模型

worker\_connections 1024; 指定每个Worker进程支持的最大连接数，数值与系统可以打开的最大文件总数（文件描述符）和主机的物理CPU和内存有关，数值不能大于系统可以打开的最大文件总数，使用cat /proc/sys/fs/file-max命令可以查看整个系统可以打开的最大文件总数包含Linux内核空间的使用，使用ulimit -SHn/n命令可以查看用户可以打开的最大文件总数，Nginx最大并发数是worker\_processes和worker\_connections数值的乘积，在设置反向代理的情况下由于三次握手四次分手建立连接，Nginx最大并发数理论上是worker\_processes\*worker\_connections/4,最大并发连接数受I/O、网络带宽和系统物理CPU内存等多方面影响，一般Nginx真实最大并发连接数小于系统可以打开的最大文件总数

}

# load modules compiled as Dynamic Shared Object (DSO)

#dso {

# load ngx\_http\_fastcgi\_module.so;

# load ngx\_http\_rewrite\_module.so;

#}

http {

include mime.types; 引入文件扩展名和文件类型映射表

default\_type application/octet-stream; 指定默认文件类型

#charset utf-8; 默认编码

server\_name\_hash\_bucket\_size 128; 服务器名称hash值表大小

client\_header\_buffer\_size 32k; 上传文件大小限制

large\_client\_header\_buffers 4 64k; 指定请求头缓存

client\_max\_body\_size 8m; 指定请求缓存

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

#access\_log "pipe:rollback logs/access\_log interval=1d baknum=7 maxsize=2G" main;

sendfile on; 指定开启高效文件传输模式，使用sendfile函数输出文件，Linux内核加载到内存数据可以直接由Linux内核控制网卡发送，默认开启

autoindex on; 开启目录列表访问，适合用于下载服务器，默认关闭

#tcp\_nopush on; 指定Nginx没有网络数据包缓存即多个软件网络数据包优先发送Nginx

tcp\_nodelay on; 防止网络阻塞

#keepalive\_timeout 0;

keepalive\_timeout 65; 指定客户端与后端服务器之间长连接缓存超时时间单位秒

#gzip on; 开启gzip压缩输出

gzip\_min\_length 1k; 最小压缩文件大小

gzip\_buffers 4 16k; 压缩缓存区

gzip\_http\_version 1.0; 压缩版本默认1.1，前端是squid2.5一般使用1.0

gzip\_comp\_level 2; 压缩等级

gzip\_types text/（plain application）/（x-javascript text）/（css application）/xml; 压缩类型，默认text/html

gzip\_vary on;

#limit\_zone crawler $binary\_remote\_addr 10m; 限制IP连接数

upstream 组服务器名 {

[server 域名或IP地址:端口号;]

}

server { 指定Nginx后端的服务器可以多个

listen 80; 监听端口

server\_name localhost; 域名，可以用空格隔开多个

#charset koi8-r;

#access\_log logs/host.access.log main;

#access\_log "pipe:rollback logs/host.access\_log interval=1d baknum=7 maxsize=2G" main;

location [=/~/~\*/^~] uri地址 {

[deny IP地址;]

allow IP地址; 按照顺序匹配IP地址

proxy\_pass http://域名IP地址或组服务器名/; 反向代理重定向到指定的完整域名或者IP地址即URL请求资源

} 指定对uri地址路径（默认指定路径及子路径下所有）[精确匹配指定路径/区分字符大小写的正则表达式模式匹配/不区分字符大小写的正则表达式模式匹配/不使用正则表达式模式匹配]的资源进行访问，优先级从大到小：精确匹配、不使用正则表达式模式、区分/不区分字符大小写的正则表达式模式、默认

location / {

root html; Web服务主目录路径

index index.html index.htm; 首页文件，使用autoindex on;开启目录列表访问，适合用于下载服务器，默认关闭

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#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;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

# another virtual host using mix of IP-, name-, and port-based configuration

#

#server {

# listen 8000;

# listen somename:8080;

# server\_name somename alias another.alias;

# location / {

# root html;

# index index.html index.htm;

# }

#}

# HTTPS server

#

#server {

# listen 443 ssl;

# server\_name localhost;

# ssl\_certificate cert.pem;

# ssl\_certificate\_key cert.key;

# ssl\_session\_cache shared:SSL:1m;

# ssl\_session\_timeout 5m;

# ssl\_ciphers HIGH:!aNULL:!MD5;

# ssl\_prefer\_server\_ciphers on;

# location / {

# root html;

# index index.html index.htm;

# }

#}

}