Lvs+keepalived工作原理及基本使用

# LVS简介

LVS是Linux Virtual Server的简写，意即Linux虚拟服务器，是一个虚拟的服务器集群系统。   
LVS采用IP负载均衡技术和基于内容请求分发技术。调度器具有很好的吞吐率，将请求均衡地转移到不同的服务器上执行，且调度器自动屏蔽掉服务器的故障，从而将一组服务器构成一个高性能的、高可用的虚拟服务器。整个服务器集群的结构对客户是透明的，而且无需修改客户端和服务器端的程序。为此，在设计时需要考虑系统的透明性、可伸缩性、高可用性和易管理性

# Keepalived原理

Keepalived是Linux下一个轻量级别的高可用解决方案，主要是通过vrrp协议，解决静态路由出现的单点故障问题，在这里简单的说就是通过冗余出可动态绑定的虚拟ip来实现高可用，一般在两台机器上配置好冗余的vip，id，以及对方ip，权重，启动后根据权重决定vip先绑定的机器，先绑定vip的机器会接收请求并回应，另一台机器对高权重机器做心跳检测，若高权重机器宕掉，该机器检测到会立即将此ip绑定到本机器

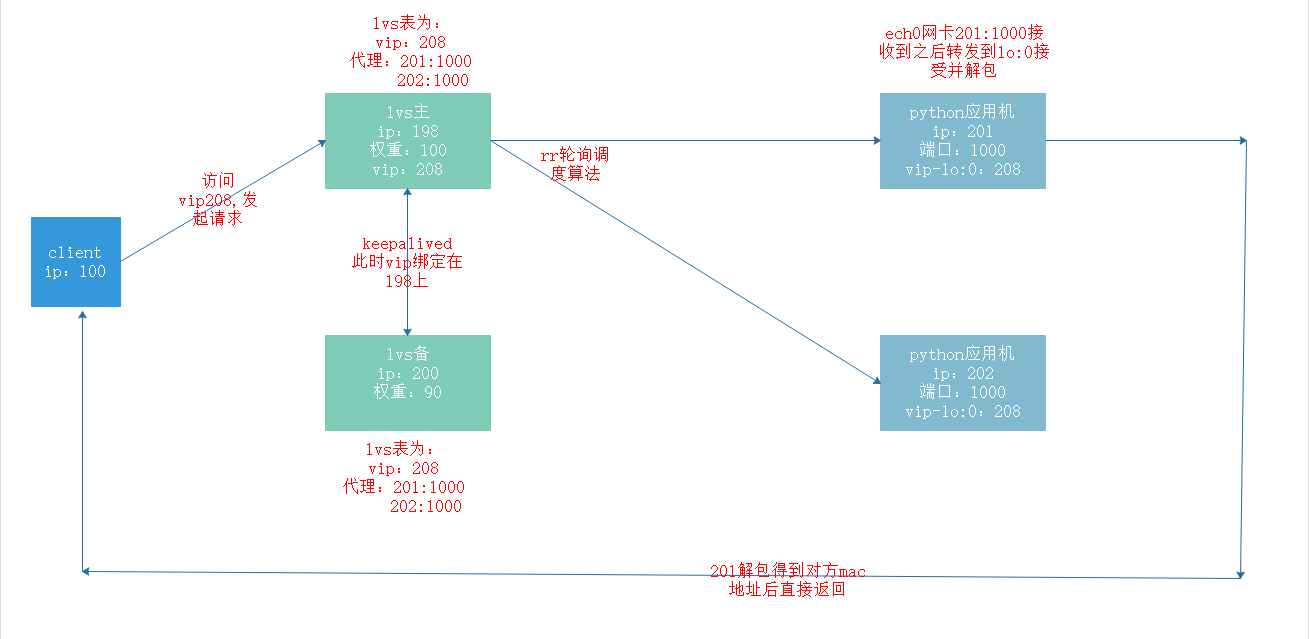
# Lvs：DR模式原理

对于我们公司来说我们用到了DR模式，而且DR及（直接路由模式）是使用最多的一种模式，较适合我们公司环境，

### 原理

DR模式是通过改写请求报文的目标MAC地址，将请求发给真实服务器的，而真实服务器响应后的处理结果直接返回给客户端用户。同TUN模式一样，DR模式可以极大的提高集群系统的伸缩性。而且DR模式没有IP隧道的开销，对集群中的真实服务器也没有必要必须支持IP隧道协议的要求。但是要求调度器LB与真实服务器RS都有一块网卡连接到同一物理网段上，必须在同一个局域网环境

### 原理图



### 原理图总结详解

1. 客户端发起一个请求到vip（虚拟ip）208的1000端口上（ps：这里添加一下arp协议解释，客户端寻找208之前会去找路由发送一个广播去询问谁是208，因为此时客户端不知道208在哪以及它的物理地址，208收到广播后回应请求，并附带自己的mac地址，路由给客户端以及208完成mac地址交换）
2. 此时lvs主198通过keepalived冗余出vip208并绑定到机器（198权重为100，所以先绑定），然后208接受请求，查询lvs转发表
3. 查询到转发表内有201，202机器的1000端口，通过rr（随机轮询调度算法）将请求转发到201机器的1000端口
4. 201机器的eth0网卡ip为201接收到请求之后，转发到lo:0网卡vip为208去解包（ps：这里解释一下lo:0的作用，这里起到的是转发作用，而且，该网卡不回应请求，且做了arp帧抑制，目的是防止该网卡回应客户端让路由发起的arp广播，导致arp混乱）
5. lo:0解开数据包各层包头获得请求的端口为1000，并交给nginx，nginx的代理再转发到内部具体python进程
6. 处理完成之后，数据包开始封装并交给lo:0网卡转发到eth0网卡，并直接返回给client，这里因为之前解开的链路层包头内含有源地址的mac地址，在网络中，mac地址即物理地址，获得后即直接寻找该机器，内网内不会通过路由，外网则需要路由去转发该请求（ps：与我们人类直接寻找对方家庭住址性质相同，至于为什么知道是在内网，因为本机器会用本机配置的子网掩码与目标地址进行比对换算，对方IP地址在解开ip包头后就已经获得，如发现在同一网段则直接发送，不在则交给网关转发）

# 安装方法

### 安装lvs+keepalived

yum install ipvsadm -y

yum install keepalived –y

# 配置文件

### Lvs

附上lvs添加脚本

#!/bin/bash

echo 1 > /proc/sys/net/ipv4/ip\_forward #开启转发

ipv=/sbin/ipvsadm

vip=10.0.1.208 #需要冗余出来的虚拟ip

rs1=10.0.1.201 #第一个被代理服务器ip

rs2=10.0.1.202 #第二个被代理服务器ip

#$ipv –C #清除转发表，慎用

$ipv -A -t $vip:9000 -s rr #添加vip并使用轮询调度

$ipv -a -t $vip:9000 -r $rs1:9000 -g #添加vip下的cip1

$ipv -a -t $vip:9000 -r $rs2:9000 –g #添加vip下的cip2

最后，查看转发表的命令为ipvsadm -ln

### Keepalived

附上配置文件（这里两台机器除权重外其他都如下配置即可）：

vrrp\_instance VI\_1 {

state MASTER

interface eth0 #绑定的真实网卡，通过该网卡冗余出对外出口vip

virtual\_router\_id 51 #此id相同则代表一对

priority 100 #权重，另一台服务器的配置文件应该设置比此机器低

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.0.1.203 #需要冗余出来的vip

}

}

virtual\_server 10.0.1.203 9000 {

delay\_loop 6

lb\_algo rr

lb\_kind NAT

nat\_mask 255.255.255.0

persistence\_timeout 50

protocol TCP

real\_server 10.0.1.198 9000 { #vip绑定的第一个机器以及端口

weight 1

TCP\_CHECK { #这里是健康检测配置

connect\_timeout 10 #连接超时时间

nb\_get\_retry 3 #重连次数

delay\_before\_retry 3 #重连间隔

connect\_port 9000 #检查端口

}

}

real\_server 10.0.1.200 9000 { #vip绑定的第二个机器以及端口

weight 1

TCP\_CHECK {

connect\_timeout 10

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 9000

}

}

}

然后启动以及查看命令

service keepalived start #启动服务

ip addr #查看ech0网口是否冗余出虚拟ip