一种基于网络地址转换的 LVS 数据转发模式①

魏振伟, 顾乃杰, 彭建章, 张颖楠

(中国科学技术大学 计算机科学技术学院, 合肥 230027)
(中国科学技术大学 安徽省计算与通信软件重点实验室, 合肥 230027)
(中国科学技术大学 中科院沈阳计算所网络与通信联合实验室, 合肥 230027)

摘 要: 研究 Linux 虚拟服务器(Linux Virtual Server, 简称 LVS)工作原理, 分析其数据转发流程. 针对 LVS 在多虚拟局域网环境下的部署应用问题, 设计实现了一种基于网络地址转换的数据转发模式 Double-NAT. Double-NAT 模式重新组织连接哈希表为双向桶结构, 使用系统分配的端口区分 IN 和 OUT 方向的数据流, 使得多 VLAN 下的数据包均能通过网络地址转换后转发. 测试结果表明, Double-NAT 数据转发模式配置简单、性能良好, 能够有效应用于多虚拟局域网环境.

关键词: Linux 虚拟服务器; 网络地址转换; 数据转发; 最大并发连接数

A NAT-Based Data Forward Method for LVS

WEI Zhen-Wei, GU Nai-Jie, PENG Jian-Zhang, ZHANG Ying-Nan

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
(Anhui Province Key Laboratory of Computing and Communication Software, Hefei 230027, China)
(USTC & SICT Network and Communication Joint Laboratory, Hefei 230027, China)

Abstract: This paper studies the working principle of Linux Virtual Server (LVS), and analyzes its data forward process. To make LVS deployable in the environment of multiple VLANs (Virtual Local Area Network), this paper designs and implements a data forward method naming Double-NAT based on Network Address Translation(NAT). The Double-NAT method organizes the connection hash table in double-direction hash bucket, and uses system allocated port to distinguish data flow of IN and OUT direction, thus data in multiple VLANs can be forwarded via network address translation. Test results show that Double-NAT data forward method makes LVS work well in multiple VLANs environment with easy configuration and good performance.

Key words: Linux virtual server; network address translation; data forward; maximum concurrent connection

1 引言

互联网应用的快速增长使得网络服务器面对的用户访问请求呈爆炸性增长, 因此服务器需要具备响应大数量级用户并发请求的能力. 单机服务器性能的物理上限使得服务器只依靠提升单台服务器性能无法从根本上满足日益增长的用户需求. 在这种背景下, 服务器集群和网络负载均衡技术[1]成为一种满足大数量级用户并发访问互联网服务的解决方案.

目前业界著名的开源负载均衡解决方案有 Haproxy[2], LVS[3]等. 其中 Linux 虚拟服务器(Linux Virtual Server, LVS)因其高性能、易扩展、配置简单、硬件要求低而被广泛应用. LVS 在 Linux 内核中实现基于 IP 层的负载均衡方案. LVS 把用户请求根据负载均衡算法转发到不同的物理服务器上, 从而将一组物理服务器构造成一个高性能的虚拟服务器[4].

在多虚拟局域网(Virtual Local Area Network, VLAN)应用环境下, 后端物理服务器基于地理位置或上层业务逻辑分布在不同的虚拟局域网中. 在这种环境下, 用户请求需要能够转发到不同虚拟局域网下的物理服务器. 目前 LVS 原有的三种数据转发模式均不

① 基金项目:"核高基"重大专项(2009ZX01028-002-003-005);高等学校学科创新引智计划(B07033)
收稿时间:2013-03-14;收到修改稿时间:2013-04-15

能很好应用于这种环境.

针对该问题, 本文设计实现了一种基于网络地址转换的数据转发模式 Double-NAT. 该模式通过修改数据包的网络地址, 使得 LVS 能够转发用户请求到任一 VLAN 中的物理服务器, 也使得 LVS 能够转发服务器响应到用户. 测试表明, Double-NAT 数据转发模式能够有效应用于多 VLAN 环境.

本文剩余部分组织如下: 第 2 节介绍背景知识, 第 3 节论述 Double-NAT 工作原理, 第 4 节详细讨论 Double-NAT 实现, 第 5 节测试, 第 6 节总结.

2 背景

2.1 LVS 和 Netfilter

LVS 在 Linux 内核网络协议栈中实现了基于 IP 层的用户请求转发. 这个内核功能模块叫做 IP 虚拟服务器(IP Virtual Server, IPVS), 它是一个基于 Netfilter 机制[5]的 IP 层数据包调度器.

Netfilter 是 Linux 内核的防火墙框架, 它在 Linux 内核网络协议栈上设置 5 个数据检测点(HOOK), 并且在每个检测点下注册数据包处理函数. 数据包在经过检测点时处理函数对目标数据包进行处理, 实现诸如包过滤、网络地址转换、防火墙等功能. 文献[6]详细分析 Netfilter 的工作原理以及在包过滤、连接跟踪等方面的应用.

IPVS 在 Netfilter 框架的 3 个监测点注册了数据包处理函数, 如图 1 所示: 在 NF_INET_LOCAL_IN 处注册 ip_vs_in, 该函数是 IPVS 入口点, 执行连接哈希表查找、连接建立、数据包改写和转发等操作; 在 NF_INET_LOCAL_OUT 处注册 ip_vs_out 和 ip_vs_forward_icmp, 前者转发 VS/NAT 模式下的响应数据包, 后者处理 ICMP 数据包; 在 NF_INET_POST_ROUTING 处注册 ip_vs_post_routing, 它对标记为 IPVS 连接的数据包执行快速发送处理.

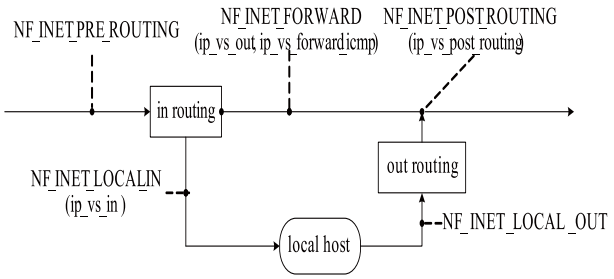


图 1 IPVS 在 Netfilter 注册的处理函数

2.2 LVS 工作原理

从体系结构上讲, LVS 集群系统主要由负载调度器(load balancer)和服务器池(server pool)组成[7], 如图 2 所示.

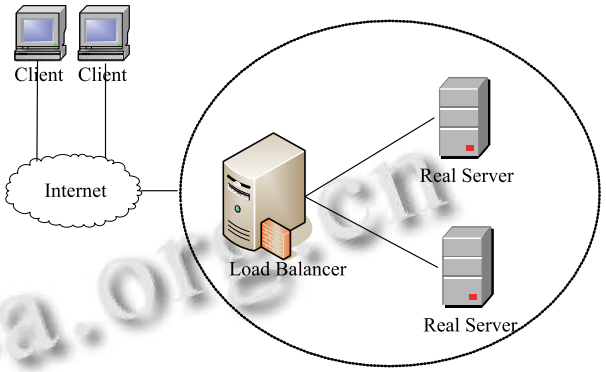


图 2 LVS 集群体系结构

负载调度器是整个服务器集群系统对外的服务接口, 它接收客户端请求, 把请求转发到服务器池中的物理服务器. 在实际部署应用中, 负载调度器配置若干虚拟服务器(Virtual Server), 向外提供服务地址. 服务器池是一组实际执行客户请求的后端物理服务器, 它们接收并响应由负载调度器转发的请求, 而这些请求实际上来自客户端.

IPVS 在转发用户请求时, 把用户请求抽象为连接, 所有连接存储在全局连接哈希表. 当新用户请求到达负载调度器时, IPVS 根据调度算法从服务器池中选择物理服务器, 建立连接, 把连接存入连接哈希表. 然后 IPVS 根据数据转发模式修改数据包, 把数据包转发到选择的后端服务器. 后端服务器接收并处理用户请求, 把响应数据包返回给用户. 当这个请求的后续数据包到达负载调度器时, IPVS 查找连接哈希表得到后端服务器, 把数据包转发到后端服务器.

2.3 LVS 数据转发模式

IPVS 的负载调度算法选择处理用户请求的后端服务器, 数据转发模式选择用户请求的数据发送方法. LVS 目前支持三种数据转发模式: VS/NAT 模式通过改写数据包的目标地址把请求转发到后端服务器; VS/TUN 模式通过 IP 隧道封装数据包, 然后把数据包转发到后端服务器; VS/DR 模式通过改写数据包的 MAC 地址把请求转发到后端服务器. VS/NAT 模式的响应数据包经过负载调度器转发给用户, VS/TUN 和 VS/DR 模式的响应数据包直接返回给用户. 文献[8]详

细讨论了 LVS 的三种数据转发模式.

在多 VLAN 环境中, 后端服务器基于地理位置或上层业务逻辑分布在不同的局域网下. 在这种环境下, LVS 需要能够转发请求数据包到不同 VLAN 下的后端服务器, 也要能够转发来自不同 VLAN 下后端服务器的响应数据包. VS/NAT 和 VS/DR 要求负载调度器和后端服务器部署在同一 VLAN 中, 而 VS/TUN 虽然可以是后端服务器跨 VLAN 部署, 但是需要后端服务器支持 IP Tunneling 协议[9]. 因此, 这三种数据转发模式都不适用于多 VLAN 环境.

为解决 LVS 在多 VLAN 环境下的部署应用问题, 淘宝公司实现了一种数据转发模式 FULLNAT[10]. 该模式在负载调度器上配置若干 IP 地址, 叫做 Local IP. IPVS 在转发请求数据包时, 把数据包的源地址转换为 Local IP. 响应数据包到达负载调度器时, IPVS 把数据包的目的地址即 Local IP 转换为负载调度器的虚拟服务器地址.

FULLNAT 模式使得 LVS 能够在多 VLAN 环境下部署应用, 但是它需要专门为负载调度器配置 Local IP. 并且 FULLNAT 模式中本地端口分配会影响到用户态网络应用进程.

针对这些问题, 本文设计实现了一种基于网络地址转换的数据转发模式 Double-NAT. Double-NAT 模式既能够有效应用于多 VLAN 环境, 又避免了 FULLNAT 模式的本地端口分配影响用户态网络进程的问题, 并且无需配置 Local IP. 综合来看, Double-NAT 较好地解决了 LVS 在多 VLAN 环境下的应用问题.

3 Double-NAT设计

Double-NAT 数据转发模式在 VS/NAT 模式的基础上, 对数据包的源 IP 地址和源端口、目的 IP 地址和目的端口均作转换, 使响应数据包的目的地址指向负载调度器. 在多 VLAN 环境下, 只要负载调度器和各 VLAN 下的后端服务器能够在网络层路由相通, IPVS 就能够实现数据包的转发. Double-NAT 模式的数据转发流程如图 3 所示.

在 Double-NAT 模式下, 用户请求到达负载调度器时, IPVS 完成数据包调度, 把数据包转发到后端服务器. 后端服务器的响应数据包到达负载调度器, IPVS 再把响应数据包转发到客户端.

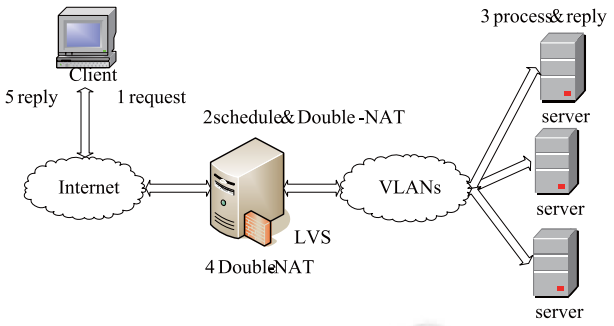


图 3　Double-NAT 数据转发流程

为便于描述, 我们把数据包的地址和端口信息描述为 ip:port 的形式: 源地址描述为 sip:sport, 目的地址为 dip:dport; 另外客户端地址为 cip:cport, 负载调度器的虚拟服务器地址为 vip:vport, 后端服务器地址为 bip:bport. Double-NAT 转发流程中数据包地址转换如图 4 所示.

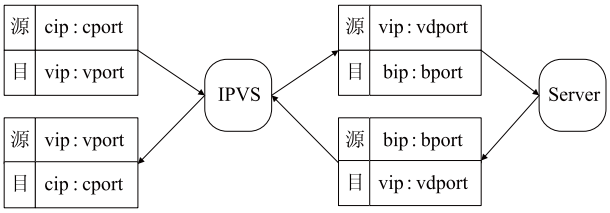


图 4　Double-NAT 地址转换过程

Double-NAT 模式下数据转发过程如下:

step 1: 客户端想 LVS 请求服务, 首个请求数据包到达负载调度器. IPVS 以数据包的地址和端口信息查找全局连接哈希表. 查找失败表明请求首次到达负载调度器, 于是调用调度函数选择后端服务器并创建连接, 把连接插入全局连接哈希表.

step 2: IPVS 调用地址和端口转换函数, 把数据包的源地址转换为虚拟服务器地址 vip, 源端口转换为向内核系统申请的端口 vdport, 目的地址转换为选定的后端服务器的地址 bip, 目的端口转换为后端服务器端口 bport, 然后重新计算数据包校验和.

step 3: IPVS 调用 IP_VS_XMIT 宏把修改后的请求数据包送往协议栈的数据转发路径, 数据包发往后端服务器.

step 4: 后端服务器接收并处理请求数据包, 将形成的响应数据包发往负载调度器.

step 5: 响应数据包到达负载调度器, IPVS 查找全

局连接哈希表得到客户端地址信息 cip:cport, 然后调用地址和端口转换函数把数据包的源地址转换为虚拟服务器地址 vip, 源端口为虚拟服务器断开 vport, 目的地址转换为客户端地址 cip, 目的端口为客户端端口 cport. 然后重新计算数据包校验和.

step 6: IPVS 调用 IP_VS_XMIT 宏把修改后的响应数据包送往协议栈的数据发送路径, 数据包发往客户端.

至此一次请求响应过程完成. 同一请求的后续数据包继续按照 step1~step6 的流程进行转发, 所不同的是在 step1 中查找全局连接哈希表时查找成功, 根据查找结果得到后端服务器地址信息, 进入 step2 进行下一步操作.

4 Double-NAT实现

4.1 连接哈希表重构

IPVS 在内核中以连接为单位转发用户请求. 一个连接就是 IPVS 在转发一次用户请求过程中所需要的信息集合, 包括数据包协议、客户端地址、虚拟服务器地址、负载均衡算法、后端服务器地址、数据包转发模式、连接超时时间等信息. 所有连接存储在全局哈希表 ip_vs_conn_tab 中. 在 IPVS 的最初实现中, ip_vs_conn_tab 采用单向桶结构, 如图 5 所示:

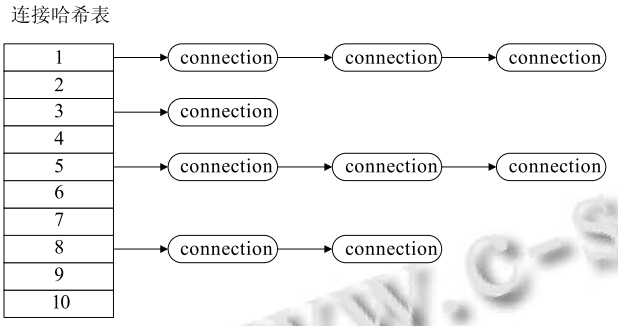


图 5 连接哈希表单向桶结构

IPVS 新建连接后, 基于数据包的协议、源地址和源端口计算连接哈希表的索引值:

$$Index = Hash(af, sip, sport) \quad (1)$$

然后把新连接插入索引值所在的双向链表中.

IPVS 创建连接时请求数据包的源地址实际上是客户端地址 cip:cport, 因此连接基于客户端地址信息存入连接哈希表. VS/NAT 模式下的响应数据包到达负载调度器时, IPVS 用数据包目的地址(即客户端地址 cip:cport)计算索引值查找哈希表. 后续请求数据包到达负载调度器时用源地址 cip:cport 查找哈希表.

在 Double-NAT 模式下, 请求数据包和响应数据包的目的地址均为虚拟服务器地址, 为了能够在查找连接哈希表时区分这两种流向的数据包, 我们采用双向桶结构重新组织连接哈希表, 如图 6 所示. 为方便描述, 我们把来自客户端的请求数据包称为 IN 方向数据包, 来自后端服务器的响应数据包称为 OUT 方向数据包.

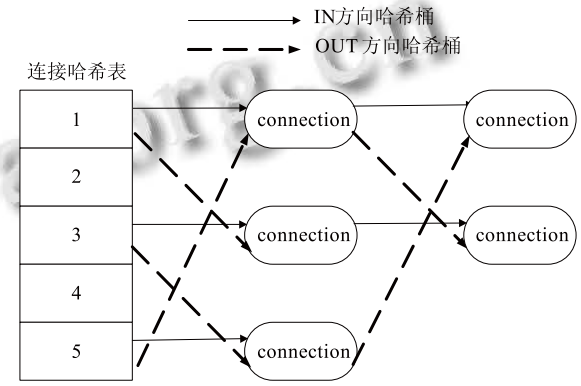


图 6 连接哈希表双向桶结构

在连接哈希表双向桶组织下, 新建连接在存入哈希表时, IPVS 分别计算 IN 方向和 OUT 方向的索引值, 然后把连接分别插入这两个索引值所在的链表中. 索引值基于数据包的协议、源地址和源端口、目的地址和目的端口计算:

$$Index = Hash\ (sip, sport, dip, dport, af) \quad (2)$$

IN 方向索引值以客户端和虚拟服务器的地址信息计算, OUT 方向索引值以后端服务器和虚拟服务器地址信息计算:

$$Index_{in} = Hash(af, cip, cport, vip, vport) \quad (3)$$

$$Index_{out} = Hash(af, bip, bport, vip, vdport) \quad (4)$$

$$Index_{out2} = Hash(af, bip, bport, cip, cport) \quad (5)$$

其中式子(4)中的 vdport 是 IPVS 为区分 IN 方向和 OUT 方向数据而向内核空闲端口哈希表申请的端口. 在实现上基于兼容性考虑, IPVS 原有三种转发模式下的 OUT 方向索引值用式子(5)计算, IN 方向索引值仍用式子(3)计算. IPVS 在 ip_vs_in 函数中查找连接哈希表时, 首先根据数据包的目的地址和端口确定数据包方向, 然后在 IN 或 OUT 哈希桶中查找连接.

4.2 系统分配端口管理

在 LVS 集群系统中, 用户请求数据包的目的地址是虚拟服务器地址 vip:vport. 在 Double-NAT 模式下,

当IPVS转发请求数据包到后端服务器时，如果把服务器的源地址改为 vip:vport，则后端服务器响应数据包的目的地址也是vip:vport. 这样IPVS便无法判断数据包是来自客户端还是服务器端. 因此 Double-NAT 模式引入向系统分配的端口 vdport 区分 IN 方向和 OUT 方向的数据流. 由 4.1 节中式子(3)和(4)可以发现，在 vport 和 vdport 不同的情况下，计算得出的 IN 方向和 OUT 方向的索引值也不相同，从而 IPVS 把同一连接插入两个不同的双向链表中.

分析 4.1 节式子(4)可以发现，在后端服务器地址 bip:bport 不同的情况下，同一 vdport 也可计算出不同的索引值，因此同一 vdport 可以被多个连接复用. 若配置IPVS的负载均衡算法为轮询算法，则IPVS转发请求到每台后端服务器的机会是均等的. 假设后端服务器一共有 N 个，那么一个 vdport 可以被复用 N 次.

IPVS 设置管理 vdport 的计数器数组 vdport[65535] 以记录每个端口的分配与复用情况. 当某个连接建立和释放时，它使用的端口 i 的计数器 vdport[i]自增 1 和自减 1. vdport[i]=N 时表明端口 i 已经被完全复用，vdport[i]=0 表明没有连接使用端口 i，释放它到系统空闲端口哈希表.

Linux 内核参数 ip_local_port_range = [low, high] 决定了系统中能够使用的端口范围. 当设置 ip_local_port_range 为[1024, 65535]时，系统可用端口数目为 6553-1024=64511. 若后端服务器总数为 N，则 IPVS 能同时保持的最大并发连接数目 Max-CC(Maximum Concurrent Connection)为:

$$Max\text{-}CC = 64511 * N \quad (6)$$

4.3 ip_vs_in 函数优化

不管是新请求数据包还是响应数据包到达负载调度器，IPVS 在 ip_vs_in 函数中都要查找连接哈希表. 原来的 ip_vs_in 函数需要两次查找哈希表以判断数据包是来自客户端还是后端服务器. 为支持 Double-NAT 转发模式和提高查找效率，我们优化了 ip_vs_in 函数，使得 IPVS 根据数据包目的地址的特征只查找一次连接哈希表就能够转发数据包. 优化后的 ip_vs_in 函数算法流程如图 7 所示.

算法首先调用 lookup_service 函数把数据包目的地址端口 dip:dport 和虚拟服务器地址端口 vip:vport 作比较. 根据比较结果判断数据流向，然后在连接哈希表的 IN 或 OUT 方向哈希桶上查找:

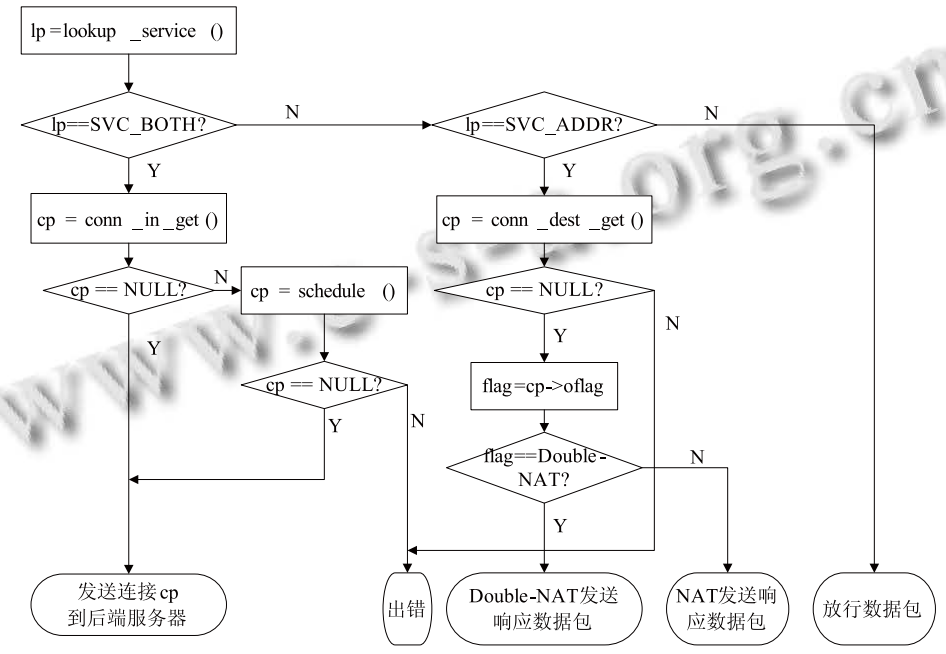


图 7　ip_vs_in 函数优化

1) dip == vip, dport == vport: 该数据包来自客户端，查找 IN 方向的哈希桶. 查找成功说明数据包是某连接的后续数据；查找失败说明是数据包是一个新请求，创建连接，把连接插入 IN 和 OUT 哈希桶中. 然后

数据包发往后端服务器.

2) dip == vip, dport != vport: 该数据包可能是 VS/NAT 模式下由负载调度器发出的用户请求的响应数据包或者 Double-NAT 模式下的响应数据包, 查找 OUT 方向哈希桶. 查找成功则根据连接中的标志位选择进入 VS/NAT 响应数据包发送路径还是 Double-NAT 响应数据包发送路径; 查找失败则进入出错处理.

3) dip != vip: 该数据包不归 LVS 系统处理, 直接返回 NF_ACCEPT 放行数据包.

5 实验

5.1 实验环境和方法

为了直观反映 Double-NAT 的转发功能和性能, 本文对 Double-NAT 数据转发模式进行了一系列功能及性能测试. 测试环境如表 1 所示:

表 1 Double-NAT 测试环境

CPU	Genuine Intel i7_2600 CPU@3.40GHz
内存	16GB
操作系统	64 位 CentOS 5.9
内核版本	Linux 2.6.35
网卡	Intel®82580EB
测试仪	IxLoad 6.10.0.1253 EA[11]

其中测试仪 IxLoad 模拟客户端和后端服务器: 客户端分布在 10.2.0.0/16 网段, 模拟 100 个客户端; 服务器平均分布在 10.1.0.0/16 和 10.3.0.0/16 网段, 每个网段根据需要模拟 3~10 台服务器. 虚拟服务器的负载均衡算法为轮询算法. 测试机的 Linux 内核参数可用端口范围 ip_local_port_range 设置为[1024, 65535].

5.1 实验结果和分析

不同的网络性能指标反映 LVS 转发性能的不同方面, 本节通过目前业界公认的几个指标综合衡量 Double-NAT 模式的转发性能.

每秒新增连接数 CPS(Connections Per Second)即 LVS 能够处理的每秒钟新增加的用户请求数目, 它反映 LVS 处理用户请求的速度. 测试结果表明 CPS 约为 22 万连接/秒左右, 如图 8 所示.

最大并发连接数 Max-CC(Maximum Concurrent Connections)反映 LVS 能够同时处理用户请求的能力. 由 4.2 节可知, 在虚拟服务器配置轮询负载均衡算法的情况下, 最大并发连接数和后端服务器总数呈线性增长关系. 测试结果表明当 N=6 时, Max-CC=387066, N=8 时 Max-CC=516088, 如图 9 所示.

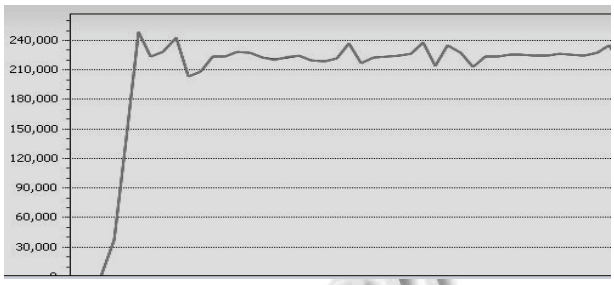
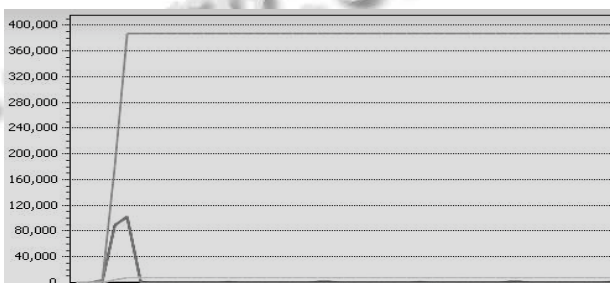


图 8 CPS 测试数据



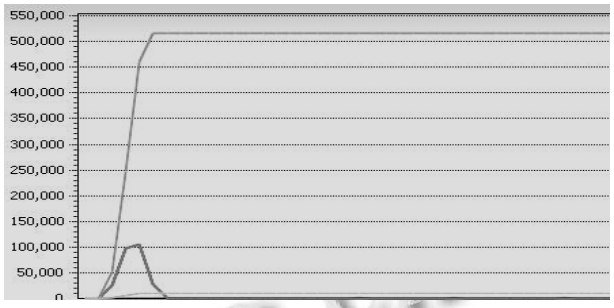


图 9 Max-CC 测试数据

数据吞吐率 Throughput 表示单位时间内通过负载调度器的网络数据量, 包括 IN 和 OUT 两个方向的数据. 它从数据量上反映 LVS 用户请求的能力. 测试表明 Throughput 平均值约为 3.3Gbps, 如图 10 所示:

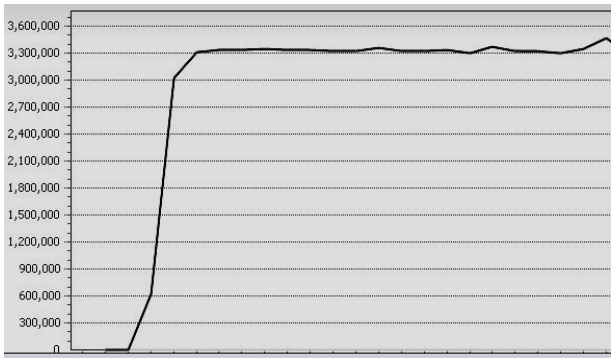


图 10 hroughput 测试数据

5.3 和 VS/NAT 的比较

从工作原理上讲, Double-NAT 模式的响应数据包经过负载调度器转发, 而 VS/NAT、VS/DR 模式的响应数据包直接返回给用户. 这使得后两者的转发性能要比 Double-NAT 好. Double-NAT 和 VS/NAT 的工作原理相似, 响应数据包都由负载调度器转发, 在性能上具有可比性. 本节通过测试 VS/NAT 的转发性能, 以比较分析二者的性能优劣. 测试环境在 Double-NAT 测试环境的基础上移除了服务器端网段 10.3.0.0/16.

在每秒新增连接数方面, 和 VS/NAT 相比, Double-NAT 在网络地址和端口转换时要多做一些操作, 这使得它的 CPS 值要稍低一些. 在当前的测试环境下, VS/NAT 的 CPS 测试值约为 22 万连接/秒, 如图 11 所示. Double-NAT 的 CPS 测试值(图 8)和它相比, 差别并不明显.

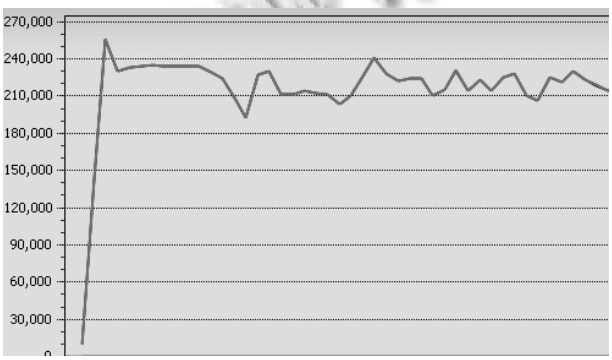


图 11　VS/NAT 模式下 CPS 测试数据

在最大并发连接数方面, VS/NAT 理论上只和负载调度器的硬件性能有关, 而 Double-NAT 则受后端服务器地址数目的线性限制. 这是 Double-NAT 相比 VS/NAT 明显不足的地方. 不过 Double-NAT 的最大并发连接数可以随着后端服务器地址的数目线性增长. 在目前的测试环境下, VS/NAT 模式的最大并发连接数可以达到 100 万, 如图 12 所示. 通过对后端服务器地址进行配置, Double-NAT 模式也可以达到这个值.

在数据吞吐率方面, Double-NAT 和 VS/NAT 一样, 都受负载调度器网卡性能的影响. 在当前的测试环境下, VS/NAT 的数据吞吐率测试值约为 3.3Gbps, 如图 13 所示.

通过以上比较可以发现, 在当前的测试环境下, Double-NAT 模式的转发性能和 VS/NAT 模式在同一个水平上.

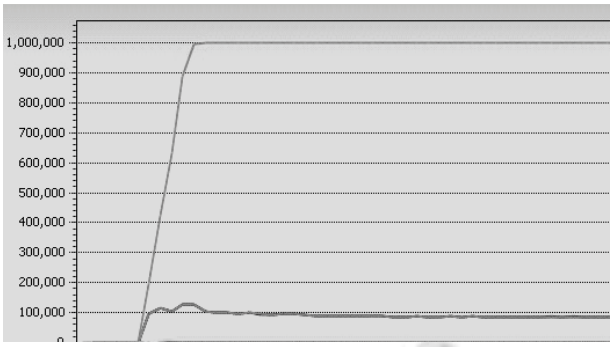


图 12　VS/NAT 模式下 CPS 测试数据

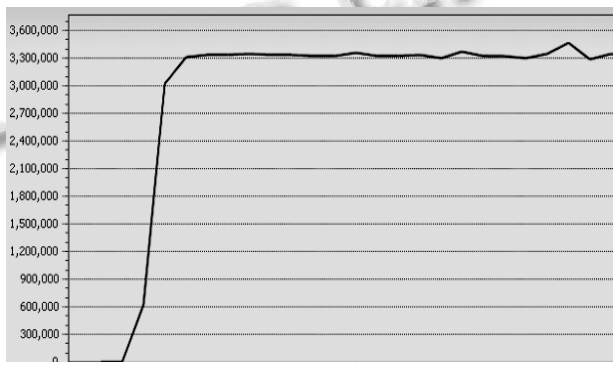


图 13　VS/NAT 模式下 Throughput 测试数据

6　总结

本文针对 LVS 在多 VLAN 场景下的部署应用问题, 在 VS/NAT 和 FULLNAT 转发模式的基础上, 设计实现了一种基于网络地址转换的数据转发模式 Double-NAT. 和 LVS 原有三种转发模式相比, Double-NAT 解决了 LVS 在多 VLAN 环境下的部署应用问题; 和 FULLNAT 转发模式相比, Double-NAT 配置简单, 并且避免了 IPVS 端口分配对用户态网络进程的干扰. 综合看来, Double-NAT 有效解决了 LVS 在多 VLAN 环境下的部署应用问题.

参考文献

1 Willy T. Making applications scalable with load balancing. http://1wt.eu/articles/2006_lb/.

2 Haproxy: The Reliable, High Performance TCP/HTTP Load Balancer. http://haproxy.1wt.eu/.

3 Zhang WS. Linux virtual server for scalable network services. Ottawa Linux Symposium.2000:1–10.

4 罗秋明,李晶.Linux 集群服务器系统 LVS 的分析与研究. 计算机时代,2006,3:6–8.

5 Duncan N. Security: IP Taxables/NetFilter-Linux's next-

(下转第 35 页)

```
else
                {
                    this.Response.Write("<script type='text/javascript' language='javascript'>alert('用户或密码错误!')</script>");                }
            }
            finally
            {
                con.Close();
            }
          }
```

除了在登录中储存会话变量，还必须为其他模块中设置一段注册判断代码，将非法的编辑页面访问行为全部转到“错误界面”中去，从而可以拦截非法登录信息管理系统的行为.

```
……
{    string a = (string)Session["usertype"];
string b = (string)Session["user"];
            if (a != "" or b != "")
            { Response.Redirect("error.htm"); }
}
```

4 结语

本文论述了基于 ASP.NET+ ACCESS 技术的农药残留检测能力验证结果上报系统的设计与实现. 本系统结构设计合理，经测试，系统运行稳定，并实现了原定的及时、准确、简便、可靠的设计目标，同时，为今后实现其他类型数据结果上报预留了改进空间，具有良好的扩展性.

参考文献

1 王璐,刘潇威,罗铭,彭祎.农产品中农药残留检测能力验证现状与展望.农产品质量与安全,2012,4:41–43.

2 张伟罡.基于 ASP.NET 技术的学校网络办公系统的设计与实现.计算机应用与软件,2012,29(11):243–247.

3 史建江,李世银,黄兴,顾军.基于 ASP.NET 的信息管理系统设计与实现.微计算机信息,2008,24(2-3):32–33,54.

4 王凤玲.基于 PHP+MYSQL 的新闻发布系统的研究与实现.计算机应用与软件,2012,29(2):234–236.

5 万敏,蒋智强.Access 数据库中操作查询探析.电脑知识与技术,2012,8(32):7621–7643.

6 夏阳,张强,陈小林.基于 ASP.Net 的电子商务网站开发与设计.计算机工程与设计,2004,25(11):2027–2029.

7 洪石丹等.ASP.NET 范例开发大全.北京:清华大学出版社,2010:5–7.

8 赵强,张红忠.基于 ASP.NET 的网站系统安全性设计与实现.计算机应用,2008,28:271–273,279.

9 卢旭,程良伦.ASP 和 ASP.NET 共享 Session 状态研究.计算机应用与软件,2009,26(6):54–56.

(上接第 24 页)

generation stateful packet filter.Sys Admin,2001,10(12):8–16.

6 王一平,韦卫.网络安全框架 Netfilter 在 Linux 中的实现.计算机工程与设计,2006,27(3):439–442.

7 章文嵩.可伸缩网络服务的研究与实现[博士学位论文].长沙:国防科技大学,2000.

8 苏命峰.三种 LVS 负载均衡模式及性能研究.自动化与信息工程,2012,32(6):17–21.

9 Simpson W. RFC 1853,IP in IP Tunneling. Network Working Group,1995.

10 IPVS FULLNAT and SYNPROXY. http: //kb.linuxvirtualserver.org/wiki/IPVS_FULLNAT_and_SYNPROXY.

11 IxLoad.http://www.ixiacom.cn/products/applications/ix load.