摘 要

随着互联网业务量的快速增长，数据流量和计算强度不断增大，当前 B/S 服务模式下，服务器端越来越成为网络业务发展的瓶颈。如何建立高可靠、高性价比、可扩展的网络服务来满足不断增长的网络需求已成为当前亟待解决的问题。基于 Linux 虚拟服务器(Linux Virtual Server)的集群技术正是解决上述问题的方法之一。

本文针对校园网内网络教学服务器存在的访问瓶颈问题，采用 Linux 虚拟服务器集群技术，利用现有的网络设备构建了一个具有负载均衡功能的 Web 服务器集群。该集群可以将访问请求自动均衡到多个服务器，同时具有高可用性和易扩展等优点。

负载均衡调度算法是 LVS 集群实现的关键，本文对 LVS 的调度算法进行了认真研究，指出其不足之处，提出并实现了一种动态反馈负载均衡算法。实验证明该算法的效率要高于加权最小连接算法。

本文主要工作如下：

1. 提出基于 LVS 的 Web 服务器集群的方案，该方案可以提高课件服务器的性能，解决课件服务器的访问瓶颈问题。

2. 研究了 LVS 集群的体系结构、工作原理和软件结构以及实现细节，重点分析了 LVS 集群中采用的四种调度算法，总结了各种调度算法的优缺点。

3. 提出并实现了一种利用后端真实服务器的实时负载信息的动态反馈负载均衡算法，相对于静态调度算法，本算法不会出现负载倾斜状况，能更好的平衡各服务器间的负载。

4. 采用 VS/NAT 模式和加权最少连接调度算法，组建了一个基于 IP 层负载均衡技术的 Web 服务器集群。

5. 在 Web 服务器集群上测试了动态反馈负载均衡算法的性能和集群系统的容错性及伸缩性，结果表明该算法能够很好的满足实际需求，具有很好的容错性和伸缩性，动态反馈调度算法极大的提高了服务器集群的性能。

关键词 LVS；集群；负载均衡；动态反馈

Abstract

Along with the rapid growth of Internet, the data flux and calculation intension grow up continually, and more and more bottlenecks would appear in the servers under the present B/S service model. It has become a problem stared in the face to build a server with high reliability, high performance/price, and scalability to meet the increasing requirements of load. In this circumstances, one solution of the above problem is the cluster based on Linux virtual server.

Aiming at solving the service bottlenecks of courseware server in campus network, the thesis built a load balancing web server cluster taking advantage of Linux virtual server and the existing device. The cluster has advantages of automatic load balancing, high usability and easy scalability.

Load balancing scheduling algorithm is the key to realizing LVS cluster. The thesis profoundly studied scheduling algorithm, pointed the weaknesses of current algorithms used in LVS cluster, and presented a dynamic feedback algorithm of load balancing. The experiments implemented in this study show the efficiency of new algorithm is superior to Weighted Least Connection Scheduling.

The main contents of this study include:

1. Brings forward a scheme to build web server cluster using LVS. The scheme can improve the performance of couserware server, solve the bottlenecks of couserware server.

2. The study of the architecture, principle, software structure and details of the implementation of LVS cluster, pay more attention to the four scheduling algorithms implemented in LVS, and gives the advantages and disadvantages of each schedule.

3. Presenting and implementing a dynamic feedback algorithm of load balancing using the real-time load information of real servers. The dynamic algorithm balances the loading better in real servers relative to static algorithm.

4. Selecting the VS/NAT model and WLC scheduling we design and construct a web server cluster based on IP layer's load balancing. Test testifies the cluster is good enough to meet the practice of web teaching requirements.

5. Implementing and testing the dynamic feedback algorithm of load balancing and on web server cluster, the result testifies it can improve the performance of the cluster greatly.

Key words LVS; Cluster; Load Balancing; Dynamic Feedback

独 创 性 声 明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京工业大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签名：[signature] 日期：2009.2

关于论文使用授权的说明

本人完全了解北京工业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵守此规定）

签名：[signature] 导师签名：[signature] 日期：2009.2

第 1 章 绪论

当今计算机技术已进入以网络为中心的计算时期。由于客户机/服务器模型的简单性、易管理性和易维护性，客户机/服务器计算模式在网上被大量采用。在九十年代中期，万维网(World Wide Web)的出现以其简单操作方式将图文并茂的网上信息带给普通大众，Web也正在从一种内容发送机制成为一种服务平台，大量的服务和应用都是围绕着Web进行[1]。这促使Internet用户剧烈增长和Internet流量爆炸式地增长。

Internet的飞速发展给网络带宽和服务器带来巨大的挑战，单一的服务器已经不能满足各种高性能计算的需求。这种情况下，人们开始寻求使用普通PC机或工作站来代替昂贵的超级计算机，于是集群技术应运而生。

1.1 课题研究背景

我校(石家庄经济学院)的网络选修课系统，是基于校园网的B/S访问模式，每学期开课十多门，选课人数700多人。目前只有1台Web服务器和1台网络存储服务器。由于教学课件全部为视频课件，传输流量很大，访问高峰时(100人以上)，服务器的工作负载明显加大，请求响应速度明显降低，有时甚至会出现服务器宕机现象。今后随着选课人数的增多这一问题将更加突出。解决办法之一是购买性能更高的新服务器，这一方法虽然能解决问题，但首先成本太高，其次是许多原有服务器仍然较新，各方面的性能指标良好，废弃不用将造成极大的资源浪费。而采用多台性能较低的服务器和PC机构建服务器集群系统恰恰是解决这类问题的较好方法。

本课题拟采用基于Linux操作系统的LVS集群技术，使用多台服务器和PC机组成服务器集群，一方面可以满足当前形势下应用系统对服务器性能的需求，另一方面可以通过给集群加入更多的服务器来满足将来不断增长的系统性能需求。服务器集群系统以其可伸缩性、高可用性、可管理性、价格低廉等优点被众多IT企业用来提供网络服务。服务器集群的构建不仅适用于网上课件点播，对于web服务、邮件服务、流媒体等大数据量访问的网络服务也同样适用。因此服务器集群系统的构建和应用对于推动现代信息技术和校园网在教学领域的应用具有积极的意义。

1.2 集群的国内外研究现状

1.2.1 集群技术的国外研究现状

集群系统采用的操作系统主要有VMS, UNIX, WINDOWS NT和LINUX。VMS CLUSTER是由美国的DEC公司开发的[2]，但它只能适用于VAX系列和ALPHA系列服务器，应用受到极大的限制。Microsoft公司从Windows2000开始增加了集群功能Wolfpack(或称MCS)，但Wolf pack目前只支持二个服务器，而且在高可用性、高可靠性和可升级性等方面与其他系统存在较大的差距，因此应用也非常受限制。

基于UNIX的集群系统由于其运行稳定、安全性好而被许多大公司所采用，如DEC, HP, SUN, IBM等公司，基于UNIX的集群系统由于它集高可靠性、高可用性和易管理性于一身，是关键业务计算机系统的理想解决方案。由于Linux的开放源代码的公共惯例，使得基于Linux的集群系统自20世纪90年代末迅速发展，并不断走向成熟，它提供了标准化的PVM, MPI消息传递机制，在PC机上提供了对高性能网络的支持，因此它发展很快。

现在出现了各种各样的Linux集群解决方案，如Turbo Linux公司推出的高可用性集群系统Turbo cluster, Red Hat推出的Piranha是基于LVS思想构建的高可用性集群，Ericsson软件工程研究中心开发的高可用性集群系统Eddie，它是能够提供较好服务质量的web服务器集群解决方案。在集群计算方面，基于Linux的MOSIX是一个高性能的集群系统。

1.2.2 集群技术的国内现状

集群服务已经成为提高网络带宽，提供更大访问流量的有效机制并被广泛推广使用。现在国内有许多新兴的软硬件产品的解决方案，联想公司在1999年推出用于分布式高性能计算的NS1000高性能集群服务器，该系统是一个四结点的系统，主要基于联想万全4500R服务器，以总体成本相对较低的设备组合，足以代替传统RISC小型机和中型机的工作，而价格仅为市场上同等性能小型机的1/2至1/4。

中软网络技术股份有限公司也推出了基于中软Linux3.0操作系统LVS虚拟服务器集群系统，进行了一定的优化，有效地提高了系统性能。该系统具有结点服务器兼容性好，集群服务器可靠性高的优点。中科红旗软件技术有限公司的Red flag Cluster Server3.0集群软件采用Linux虚拟服务器技术，提供了基于Web的集群

管理工具，具有很强的易用性。在集群内，除使用NFS为网络文件系统外，还可使用红旗Cluster所带的Coda网络文件系统。集群内还可采用SAN、NAS等分布式存储方案。朗新公司也推出了类似于Turbocluster的高可用性集群系统Longshine Cluster Server[3]。

在我国的科研院所中，在集群系统、网络计算及分布式处理方面研究较多的单位有：中国科学研究院、清华大学、国防科学技术大学、南京大学、武汉大学等等。

1.3 主要研究内容

本文的主要研究内容是利用LVS集群技术构建Web服务器集群，解决校园网内课件服务器访问瓶颈问题。首先介绍了LVS集群的体系结构、工作原理和软件结构及其实现细节，然后对LVS的调度算法进行了重点研究，针对当前的静态调度算法不能反映后端服务器的真实负载问题，提出了一种动态反馈的负载均衡算法，该算法可以根据后端服务器的真实负载进行调度。最后采用VS/NAT模式和WLC算法组建了一个实际的Web服务器集群，以校园网环境下的网络教学平台为应用需求对集群进行了测试，结果表明该集群和动态反馈算法能够极大提高集群的性能并满足课件服务器的访问需要。

1.4 论文章节安排

第一章 绪论。介绍了本文的研究背景、集群技术的国内外研究现状，本文的研究内容以及章节安排。

第二章 LVS集群和负载均衡介绍。详细介绍了负载均衡技术和LVS集群的体系结构、IP均衡技术、软件结构及其实现细节。

第三章 动态反馈负载均衡算法的设计和实现。重点研究了LVS内核中的四种调度算法，分析了其不足之处，提出了一种动态反馈负载均衡算法并给出了该算法的实现步骤。

第四章 Web 服务器集群系统的组建与测试。给出了在校园网环境下构建LVS 服务器集群系统的详细过程，并通过测试工具对改进算法的性能和集群系统的容错性和伸缩性进行了测试和分析。

结论。对全文工作进行小结，提出本论文的工作结果，并对今后的工作进行了展望。

最后，给出了参考文献及致谢。

第 2 章 LVS 集群和负载均衡介绍

2.1 集群系统概述

集群系统是高性能网络或者LAN进行物理连接的计算机的集合[4]。集合里的计算机又叫做节点(Nodes)，它们是一些完整的独立的计算机系统。集群中的节点可以是服务器，也可以是工作站，可以是PC，也可以是大型机甚至是MPPs。集群系统中的各个节点在保持本身计算机系统完备性的同时，还应该具有另一个更为重要的特征，即各个节点必须能够在一起协同工作，形成一个单一的、集成的系统资源。典型的集群系统的结构模型[5]如图2-1。

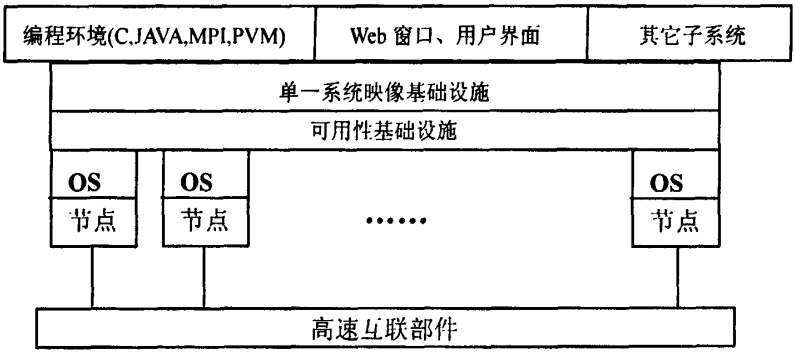

图2-1 典型的集群结构模型

Figure2-1 representative model of cluster's framework

从图2-1中可以看出来集群系统结构的特点：

集群节点：每一个节点是一个完整的计算机。这隐含着完备的计算机系统，以及相应完备的外围设备。此外，在每一个节点上都驻留着一个完整的，标准的操作系统。每一个节点上允许有一个或者多个处理器。但是它们只能有一个操作系统的映像。

单一系统映像(Single System Imgae,SSI)：整个集群系统是一个单一的计算处理资源，在这一点上集群系统和分布式系统有区别。集群系统是通过SSI技术来实现单一资源的特征。SSI技术实现了集群的有效管理和简单使用。尽管到现在为止，大多数的集群产品还不能提供完整的SSI服务，但是是花费较低的解决方案。

更好的性能：集群系统可以在很多的服务领域提供更好的性能。比如一个集群系统可以作为一个超级Web服务器，或者一个超级的数据库服务器。如果集群系统中的每一个节点可以支持N个客户，那么有M个节点的集群系统就可以支持M*N个客户。另外集群技术在并行处理领域，还可以缩短单个任务的执行时间。

更好的灵活性和可伸缩性：集群系统可以根据实际情况灵活地改变整个系统地配置，如增加或者减少节点等。这种平滑的可伸缩性不仅表现在系统规模的可伸缩性，还表现在技术，服务的可伸缩。

集群系统概念的提出带来了许多的优点，同时也带来了许多的挑战。其最主要的优点是易用性，高可用性，可伸缩性和良好的性价比。

易用性：因为集群系统的单个节点仍旧是传统的平台，所以用户可以在他们平时就很熟悉的环境下面开发和运行应用程序。同时，这也可以让许多现有的程序可以不加以修改地运行在处理能力强大的集群系统平台上，非常有利于保护用户已有的软件投资。

高可用性：高可用性包括可靠性和好的可用性等。在传统的系统中，如大型机和容错系统，通常是以高费用为代价来提供高可用性。相反，在集群系统中，却是用低费用的组件来提供较高的可用性。集群系统可用性的实现，其关键技术是开发共享组件可用的软件。

可伸缩性：一个集群系统的处理能力可以简单地通过增加节点来加强。同时，集群的可伸缩性是多面的。包括资源的可伸缩性、应用的可伸缩性和技术的可伸缩性等。SMP提供了处理器的可伸缩性，在集群系统中的可伸缩性可以是计算机的各个组件，如处理器、硬盘、内存或者I/O设备以及软件组件等。

良好的性能价格比：集群系统良好的性能价格比是它受到人们青睐的重要因素，它可以把一些廉价系统组合在一起协同工作，在总体上的性能却可以超过大型机甚至巨型机。同时，集群技术可以保护用户在原有设备上的硬件投资，用户可以用新旧设备组合起来成为一个集群，达到提供更高的性能的目的。

集群系统从不同的角度，有不同的分类：

从硬件构架看，集群系统可以分为工作站集群(Cluster of Workstation,COW)、大规模并行处理机(MPP)、对称多处理机(SMP)、分布式异构计算集群(典型是GIRD)；而从功能角度看，可以把集群系统分为三大类：高可用性集群、高性能计算集群、负载均衡集群[2]。

高可用性集群的出现是为了使集群的整体服务尽可能可用，以便考虑计算硬件和软件的容错性。如果高可用性集群中的主节点发生了故障，那么这段时间内将由备份节点代替它。备份节点通常是主节点的镜像，所以当它代替主节点时，它可以完全接管其身份，并且因此使系统环境对于用户是一致的。高可用性集群的设计思想就是要最大限度地减少服务中断时间。

高性能计算集群，通常这种集群主要是为解决复杂的科学计算问题，致力于提供单个计算机所不能提供的强大的计算能力。高性能计算是计算机科学的一个分支，它致力于开发超级计算机，研究并行算法和开发相关软件。

高性能计算主要研究两类问题：大规模科学计算问题以及存储和处理海量数

据问题。高性能计算集群就是采用集群技术来研究高性能计算，它不使用昂贵的并行超级计算机，而是通过以太网或其他网络连接的多个计算节点和管理节点构成整个集群系统。

负载均衡集群，这种集群追求的不是高速的计算能力，而是快速的事务处理和响应能力。它能够在多节点之间分发网络或计算处理负载，从而提供和节点个数成正比的负载能力。这种集群是随着Internet的发展和网络负荷增加而产生的。负载均衡集群往往也具有一定的高可用性特点，LVS集群就属于这种类型。本文所采用的LVS集群就属于负载均衡集群。

2.2 负载均衡技术

2.2.1 负载均衡简介

负载均衡技术是影响集群系统性能的关键技术之一。由于大规模用户对信息服务器访问的自相似特征表现为很强的突发性，即大量连接请求在短时间内到来，如果不采取任何措施，致使有些节点机处于重载时，另有一些却处于轻载或闲置状态，这种负载的不平衡直接影响集群系统的吞吐率与反应速度。这就有必要在集群系统中，通过任务有效地分配和调度做到负载均衡，以提高系统性能。

集群系统需要解决的关键问题是系统内部并行计算的负载均衡问题。负载均衡有两方面的含义[6]：

(1)大量的并发访问或数据流量分担到多台设备上分别处理，减少用户等待响应时间。

(2)单个重负载的运算分担到多台节点设备上做并行处理，每个节点设备处理结束后，将结果汇总，返回给用户，系统处理能力得到大幅提高。

由此可见，负载均衡就是通过重新分布系统负载，使各主机间负载达到相对均衡，从而降低任务的响应时间，提高系统资源的利用率，使系统的性能得以提高。

一个负载均衡系统应具有以下特性：

(1)高性能：负载应该比较均衡的分布在集群节点上，而且集群的规模和性能应该接近线性增长；

(2)可伸缩性：当服务的负载增长时，系统能被扩展来满足需求，且不降低服务质量；

(3)高可用性：尽管部分硬件和软件会发生故障，整个系统的服务必须是每天24小时，每星期7天可用的；

(4)可管理性：整个系统可能在物理上很大，但应该容易管理；

(5)相对的成本/性能优势：整个系统实现是经济的、易支付的，而且随着系统的增长，成本的增长必须是经济的。

2.2.2 负载均衡常用技术

负载均衡实现技术主要有以下四类[7]：

(1)基于DNS的负载均衡技术

通过DNS服务中的随机名字解析来实现的。在DNS服务器中，可以为多个不同的地址配置同一个名字，而最终查询这个名字的客户机将在解析这个名字时得到其中的一个地址。因此，对于同一个名字，不同的客户机可以得到不同的地址，也就可以访问到不同地址的服务器，从而大致达到负载均衡的目的。

这种方法有几个问题：第一，DNS是一个分布式系统，是按照一定的层次结构组织的；第二，用户机器会缓冲从名字到IP地址的映射，而不受TTL值的影响，用户的访问请求会被送到同一台WEB服务器上；第三，系统的可靠性和可维护性差。

(2)基于客户端的解决方法

基于客户端的解决方法需要每个客户程序都有一定的服务器集群的知识，进而把以负载均衡的方式将请求发到不同的服务器。由于需要每个客户程序都有服务器集群的知识，因此很不利于推广利用。

(3)基于应用层负载均衡技术

多台服务器通过高速的互联网络连接成一个集群系统，在前端有一个基于应用层的负载调度器。当用户访问请求到达调度器时，请求会提交给作负载均衡调度的应用程序，分析请求，根据各个服务器的负载情况，选出一台服务器，重写请求并向选出的服务器访问，取得结果后，再返回给用户。使用这种技术，系统处理开销特别大，致使系统的伸缩性有限，另外基于应用层的负载均衡调度器对于不同的应用，需要写不同的调度器，适用性不强。

(4)基于IP层负载均衡调度的解决方法

用户通过虚拟IP地址(Virtual IP Address)访问服务时，访问请求的报文会到达负载调度器，由它进行负载均衡调度，从一组真实服务器选出一个，将报文的目标地址Virtual IP Address改写成选定服务器的地址，报文的目标端口改写成选定服务器的相应端口，最后将报文发送给选定的服务器。真实服务器的回应报文经过负载调度器时，将报文的源地址和源端口改为Virtual IP Address和相应的端口，再把报文发给用户。在这四种算法中，基于IP层的负载均衡调度算法的效率是最高的[8]。

2.2.3 负载均衡算法介绍

负载均衡算法可以分为静态、动态和自适应算法三类[9]。

(1)静态分配算法是根据已知的有关任务的信息，通过某个策略来确定任务的分配。它不使用系统状态信息来决定负载的分布。静态负载分配算法简单，使用方便，但是并行性差，只是对某些特定的应用有较高的效率。

(2)动态负载分配算法是通过分析集群系统的实时负载信息，动态的将任务在各处理机之间进行分配和调整，以消除系统中负载分布的不平衡性。与静态算法比较，动态算法更加灵活、高效，而且在大多数情况下，动态算法要比静态算法性能提高大约30%。但是由于它必须收集、存储并分析状态信息，因此动态负载算法会产生比静态算法更多的系统开销。然而，这种开销常常可以被提高的计算机效率所抵消掉。因此动态负载分配算法成为目前研究的重点[10-19]。

动态负载均衡算法按照集中程度分为集中式、分布式、集中分布式3种。

在集中式负载均衡的系统中，各节点收集本地负载信息，并以一定时间间隔向控制节点报告，控制节点根据全局信息做出任务的分配[10]。这种方式的好处是能够以较少的开销收集全局信息，挑出最佳节点执行任务，并且可以扩充到较大的网络计算系统。LVS集群的负载分配算法就属于这种类型，但是，LVS中的各真实服务器并不收集本地负载信息，负载信息的收集和任务的分派都是由控制结点来完成。

在分布式负载平衡系统中，每个节点保存相邻节点或系统中部分节点的负载信息，相互合作做出各自的负载分配。这种策略实现比较简单，并经过一段时间后，可以选择到较合适的节点执行任务。缺点是不能获得最佳节点分配负载，很难扩展到具有成千上万个节点的集群系统中。

集中分布式是根据集中式、分布式方法的优缺点结合而成的一种负载平衡算法。由一个控制结点负责收集信息，然后向集群中的所有结点广播负载信息，每个结点根据收到的负载信息做出各自的负载分配。

(3)自适应负载均衡算法是一类特殊的动态算法，它可以通过动态改变参数、策略来调整自身的行为，以适应正在变化的系统状态。例如，某种负载均衡算法在A情况下比较有效，而另一种算法在B情况下有效，自适应算法能够根据系统状态的变化来选择合适的算法。因此，自适应算法比动态负载分配算法更为复杂，研究的难度更大，目前实际上应用较少。

2.3 LVS 集群的体系结构

Linux 虚拟服务器自由软件项目是由章文嵩博士带领，创建于 1998 年 5 月，

致力于 LVS 虚拟服务器开发工作，这项基于 LVS 的集群技术 IPVS (Based IP Virtual Server)，是一种在 Linux 操作系统上基于 IP 层的负载均衡调度技术，它在操作系统核心层上，将来自 IP 层的 TCP/UDP 请求均衡地转移到不同的服务器，从而将一组服务器构成一个虚拟服务器。该项目的目标是在 Linux 操作系统下实现一个高性能、高可用性的服务器，它具有很好的可伸缩性(Scalability)、可靠性(Reliability)和可管理性(Manageability)。目前，LVS 项目仍然处于进一步的开发和完善中，并且提供了一个实现可伸缩网络服务的 LVS 框架。

虚拟服务器构建于实际的服务器集群之上，用户看不到提供服务的多台真实服务器，而只能看见一台作为负载均衡器的服务器。真实的服务器通过高速局域网或地理上分散的广域网连接。真实服务器的前端是一台负载均衡器(Load Balancer,LB)，它将用户的请求调度到真实服务器上完成，客户访问集群系统提供的网络服务就像访问一台高性能、高可用的服务器一样[20]。

虚拟服务器集群的体系结构如图 2-2，一组服务器通过高速的局域网或者地理分布的广域网相互连接，在它们的前端有一个负载均衡器。负载均衡器能无缝地将网络请求调度到真实服务器上，从而使得服务器集群的结构对客户是透明的，客户访问集群系统提供的网络服务就像访问一台高性能、高可用的服务器一样。客户程序不受服务器集群的影响不需作任何修改。系统的伸缩性通过在服务集群中透明地加入和删除一个节点来达到，通过检测节点或服务进程故障和正确地重置系统达到高可用性。由于负载调度技术是在 Linux 内核中实现的，所以称之为 Linux 虚拟服务器，简称为 LVS[21]。

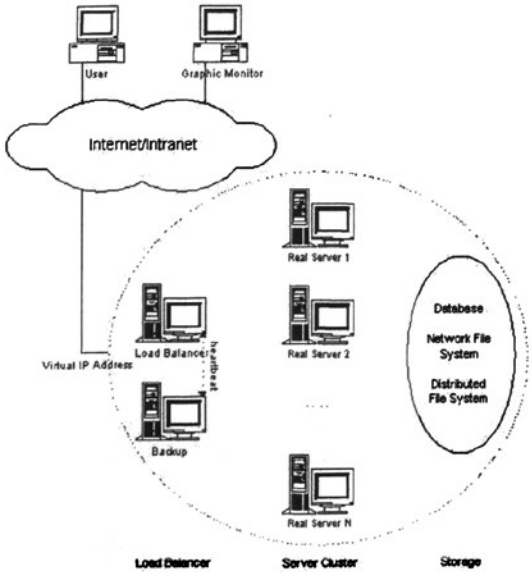

图 2-2 LVS 集群的体系结构

Figure2-2 Architecture of LVS Cluster

一般来说，LVS 集群采用三层结构，其体系结构如图 2-1 所示，三层主要组

成部分为[22]：

负载调度器（load balancer），它是整个集群对外面的前端机，负责将客户的请求发送到一组服务器上执行，而客户认为服务是来自一个 IP 地址(虚拟 IP 地址)上的。

服务器池（server pool），是一组真正执行客户请求的服务器，执行的服务有 WEB、MAIL、FTP 和 DNS 等。

共享存储（shared storage），通常是数据库、网络文件系统或者分布式文件系统。服务器结点需要动态更新的数据一般存储在数据库系统中，同时数据库会保证并发访问时数据的一致性，静态的数据可以存储在网络文件系统(如 NFS/CIFS)中。分布式文件系统可为各服务器提供共享的存储区，它们访问分布式文件系统就像访问本地文件系统一样，同时分布式文件系统可提供良好的伸缩性和可用性。

调度器是服务器集群系统的唯一入口点（Single Entry Point），它可以采用 IP 负载均衡技术、基于内容请求分发技术或者两者相结合。在 IP 负载均衡技术中，需要服务器池拥有相同的内容提供相同的服务。当客户请求到达时，调度器只根据服务器负载情况和设定的调度算法从服务器池中选出一个服务器，将该请求转发到选出的服务器，并记录这个调度；当这个请求的其他报文到达，也会被转发到前面选出的服务器。在基于内容请求分发技术中，服务器可以提供不同的服务，当客户请求到达时，调度器可根据请求的内容选择服务器执行请求。因为所有的操作都是在 Linux 操作系统核心空间中将完成的，它的调度开销很小，所以它具有很高的吞吐率。

负载调度器、服务器池和共享存储系统通过高速网络相连接，如 100Mbps 交换网络、Myrinet 和 Gigabit 网络等。使用高速的网络，主要为避免当系统规模扩大时互联网络成为整个系统的瓶颈。

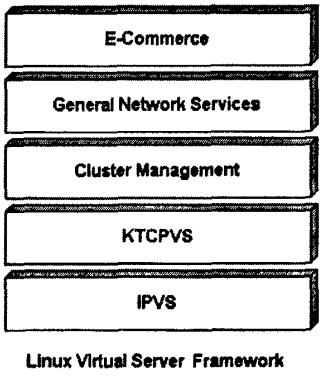

图 2-3 LVS 集群服务框架

Figure2-3 Framework of LVS Cluster's Service

目前，LVS 项目已提供了一个实现可伸缩网络服务的 LVS 框架，如图 2-3。

在 LVS 框架中，提供了含有三种 IP 负载均衡技术的 IP 虚拟服务器软件 IPVS、基于内容请求分发的内核 Layer-7 交换机 KTCPVS 和集群管理软件。可以利用 LVS 框架实现高可伸缩的、高可用的 Web、Cache、Mail 和 Media[23]等网络服务。在此基础上，可以开发支持庞大用户数的、高可伸缩的、高可用的电子商务应用。

2.4 LVS 集群的 IP 负载均衡技术

用户通过 VIP(Virtual IP Address 虚拟 IP 地址)访问服务时，访问请求的报文会到达负载调度器，由它进行负载均衡调度，从一组真实服务器选出一个，将报文的目标地址 VIP 改写成选定服务器的地址，报文的目标端口改写成选定服务器的相应端口，最后将报文发送给选定的服务器。真实服务器的回应报文经过负载调度器时，将报文的源地址和源端口改为 VIP 和相应的端口，再把报文发给用户。在已有的 IP 负载均衡技术中，主要有通过网络地址转换(Network Address Translation)将一组服务器构成一个高性能的、高可用的虚拟服务器，称之为 VS/NAT 技术(Virtual Server via Network Address Translation)。通过 IP 隧道技术实现虚拟服务器的方法 VS/TUN(Virtual Server via IP Tunneling)和通过直接路由技术实现虚拟服务器的方法 VS/DR(Virtual server via Direct Routing)，它们可以极大地提高系统的伸缩性[24]。

2.4.1 网络地址转换（VS/NAT）

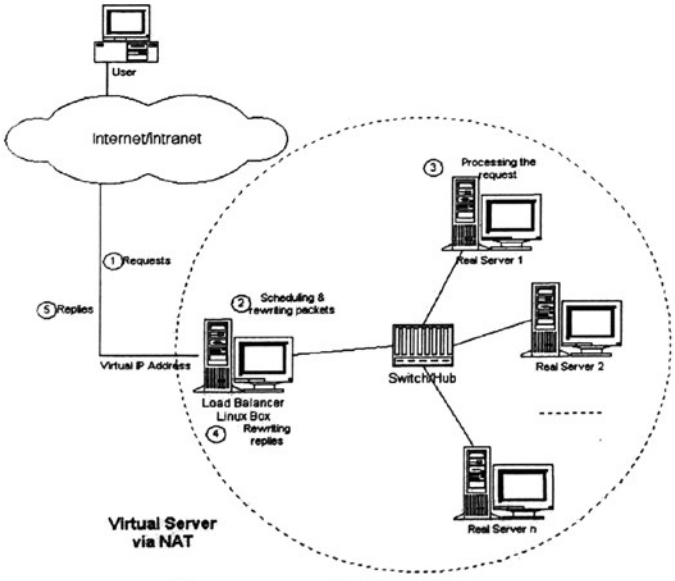

图 2-4 VS/NAT 的体系结构

Figure2-4 Architecture of VS/NAT

由于 IPv4 中 IP 地址空间的日益紧张和安全方面的原因，很多网络使用保留 IP 地址(如 192.168.0.0/255.255.0.0)。这些地址不在 Internet 上使用，而是专门为内部网络预留的。当内部网络中的主机要访问 Internet 或被 Internet 访问时，就需要采用网络地址转换，将内部地址转化为 Internet 上可用的外部地址。NAT 的工作原理是报文头（目标地址、源地址和端口等）被正确改写后，客户相信它们连接一个 IP 地址，而不同 IP 地址的服务器组也认为它们是与客户直接相连的。由此，可以用 NAT 方法将不同 IP 地址的并行网络服务变成在一个 IP 地址上的一个虚拟服务。

VS/NAT 的体系结构如图 2-4。在一组服务器前有一个调度器，它们是通过 Switch/HUB 相连接的。这些服务器提供相同的网络服务、相同的内容，即不管请求被发送到哪一台服务器，执行结果是一样的。服务的内容可以复制到每台服务器的本地硬盘上，可以通过网络文件系统（如 NFS）共享，也可以通过一个分布式文件系统来提供。

客户通过 Virtual IP Address（虚拟服务的 IP 地址）访问网络服务时，请求报文到达调度器，调度器根据连接调度算法从一组真实服务器中选出一台服务器，将报文的目标地址 Virtual IP Address 改写成选定服务器的地址，报文的目标端口改写成选定服务器的相应端口，最后将修改后的报文发送给选出的服务器。同时，调度器在连接 Hash 表中记录这个连接，当这个连接的下一个报文到达时，从连接 Hash 表中可以得到原选定服务器的地址和端口，进行同样的改写操作，并将报文传给原选定的服务器。当来自真实服务器的响应报文经过调度器时，调度器将报文的源地址和源端口改为 Virtual IP Address 和相应的端口，再把报文发给用户。这样，客户所看到的只是在 Virtual IP Address 上提供的服务，而服务器集群的结构对用户是透明的。

使用 NAT 技术，有利于保障服务器的安全，也可以节省部分 IP 地址的占用。由上述流程可知报文在负载均衡器处被处理了两次，相比后述的 VS/TUN 和 VS/DR 方式，这将明显加重负载均衡器的负担。

2.4.2 IP 隧道（VS/TUN）

在 VS/NAT 的集群系统中，请求和响应的数据报文都需要通过负载调度器，当真实服务器的数目在 10 台和 20 台之间时，负载调度器将成为整个集群系统的新瓶颈。大多数 Internet 服务都有这样的特点：请求报文较短而响应报文往往包含大量的数据。如果能将请求和响应分开处理，即在负载调度器中只负责调度请求而响应直接返回给客户，将极大地提高整个集群系统的吞吐量。

IP 隧道是将一个 IP 报文封装在另一个 IP 报文的技术，这可以使得目标为一

个 IP 地址的数据报文能被封装和转发到另一个 IP 地址。IP 隧道技术亦称为 IP 封装技术（IP encapsulation）。IP 隧道主要用于移动主机和虚拟私有网络（Virtual Private Network），在其中隧道都是静态建立的，隧道一端有一个 IP 地址，另一端也有唯一的 IP 地址。

利用 IP 隧道技术将请求报文封装转发给后端服务器，响应报文能从后端服务器直接返回给客户。但在这里，后端服务器有一组而非一个，所以不可能静态地建立一一对应的隧道，而是动态地选择一台服务器，将请求报文封装和转发给选出的服务器。这样，我们可以利用 IP 隧道的原理将一组服务器上的网络服务组成在一个 IP 地址上的虚拟网络服务。VS/TUN 的体系结构如图 2-5，各个服务器将 VIP 地址配置在自己的 IP 隧道设备上。

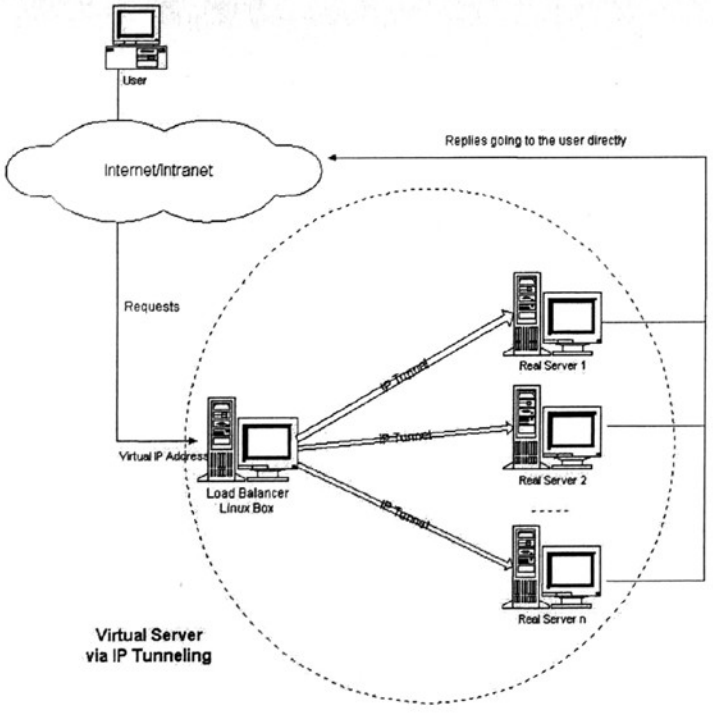

图 2-5 VS/TUN 的体系结构

Figure2-5 Architecture of VS/TUN

VS/TUN 的连接调度和管理与 VS/NAT 中的一样，只是它的报文转发方法不同。调度器根据各个服务器的负载情况，动态地选择一台服务器，将请求报文封装在另一个 IP 报文中，再将封装后的 IP 报文转发给选出的服务器；服务器收到报文后，先将报文解封获得原来目标地址为 VIP 的报文，服务器发现 VIP 地址被配置在本地的 IP 隧道设备上，所以就处理这个请求，然后根据路由表将响应报文直接返回给客户。

根据缺省的 TCP/IP 协议栈处理，请求报文的目标地址为 VIP，响应报文的源地址肯定也为 VIP，所以响应报文不需要作任何修改，可以直接返回给客户，

客户认为得到正常的服务，而不会知道究竟是哪一台服务器处理的。

2.4.3 直接路由（VS/DR）

跟 VS/TUN 方法相同，VS/DR 利用大多数 Internet 服务的非对称特点，负载调度器中只负责调度请求，而服务器直接将响应返回给客户，可以极大地提高整个集群系统的吞吐量。

VS/DR 的体系结构如图 2-6，调度器和服务器组都必须在物理上有一个网卡通过不分断的局域网相连，如通过高速的交换机或者 HUB 相连。VIP 地址为调度器和服务器组共享，调度器配置的 VIP 地址是对外可见的，用于接收虚拟服务的请求报文；所有的服务器把 VIP 地址配置在各自的 Non-ARP 网络设备上，它对外面是不可见的，只是用于处理目标地址为 VIP 的网络请求。

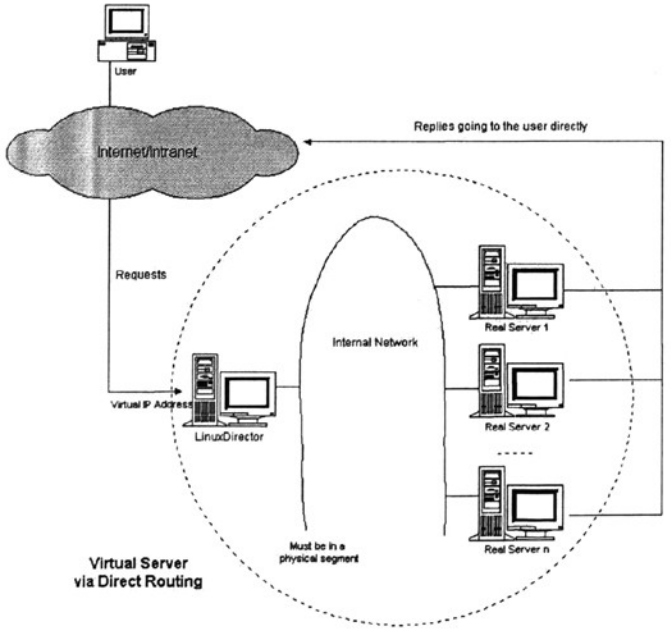

图 2-6 VS/DR 的体系结构

Figure2-6 Architecture of VS/DR

VS/DR 的连接调度和管理与 VS/NAT 和 VS/TUN 中的一样，它的报文转发方法又有不同，将报文直接路由给目标服务器。在 VS/DR 中，调度器根据各个服务器的负载情况，动态地选择一台服务器，不修改也不封装 IP 报文，而是将数据帧的 MAC 地址改为选出服务器的 MAC 地址，再将修改后的数据帧在与服务器组的局域网上发送。因为数据帧的 MAC 地址是选出的服务器，所以服务器肯定可以收到这个数据帧，从中可以获得该 IP 报文。当服务器发现报文的目标地址 VIP 是在本地的网络设备上，服务器处理这个报文，然后根据路由表将响应报文直接返回给客户。

在 VS/DR 中，根据缺省的 TCP/IP 协议栈处理，请求报文的目标地址为 VIP，响应报文的源地址肯定也为 VIP，所以响应报文不需要作任何修改，可以直接返回给客户，客户认为得到正常的服务，而不会知道是哪一台服务器处理的。

2.4.4 三种负载均衡技术比较

(1) VS/NAT

VS/NAT 的优点是服务器可以运行任何支持 TCP/IP 的操作系统，它只需要一个 IP 地址配置在调度器上，服务器组可以用私有的 IP 地址。缺点是它的伸缩能力有限，当服务器结点数目升到 20 时，调度器本身有可能成为系统的新瓶颈，因为在 VS/NAT 中请求和响应报文都需要通过负载调度器。

对于那些将 IP 地址或者端口号在报文数据中传送的网络服务，需要编写相应的应用模块来转换报文数据中的 IP 地址或者端口号。这会带来实现的工作量，同时应用模块检查报文的开销会降低系统的吞吐率。

(2) VS/TUN

在 VS/TUN 的集群系统中，负载调度器只将请求调度到不同的后端服务器，后端服务器将应答的数据直接返回给用户。这样，负载调度器就可以处理大量的请求，它甚至可以调度百台以上的服务器（同等规模的服务器），而它不会成为系统的瓶颈。即使负载调度器只有 100Mbps 的全双工网卡，整个系统的最大吞吐量可超过 1Gbps。所以，VS/TUN 可以极大地增加负载调度器调度的服务器数量。VS/TUN 调度器可以调度上百台服务器，而它本身不会成为系统的瓶颈，可以用来构建高性能的超级服务器。

VS/TUN 技术对服务器有要求，即所有的服务器必须支持“IP Tunneling”或者“IP Encapsulation”协议。目前，VS/TUN 的后端服务器主要运行 Linux 操作系统。因为“IP Tunneling”正成为各个操作系统的标准协议，所以 VS/TUN 应该会适用运行其他操作系统的后端服务器。

(3) VS/DR

跟 VS/TUN 方法一样，VS/DR 调度器只处理客户到服务器端的连接，响应数据可以直接从独立的网络路由返回给客户。这可以极大地提高 LVS 集群系统的伸缩性。

跟 VS/TUN 相比，这种方法没有 IP 隧道的开销，但是要求负载调度器与实际服务器都有一块网卡连在同一物理网段上，服务器网络设备（或者设备别名）不作 ARP 响应，或者能将报文重定向（Redirect）到本地的 Socket 端口上。

2.5 LVS 软件结构和实现

LVS 软件的核心是运行在负载均衡器上的 IPVS，它使用基于 IP 层的负载均衡方法[25-26]，IPVS 的总体结构如图 2-7，它主要由 IP 包处理、负载均衡算法、系统配置与管理三个模块及虚拟服务器与真实服务器链表组成。IPVS 的源码组织得比较好，以上几个模块分别处于相对比较独立的源码文件中。其中 IP 包处理代码主要在 ip_vs_ core.c 与 ip_vs_conn.c 文件中，各种调度算法以内核模块形式存在，它的管理主要在 ip_vs_sched.c 中实现，配置与管理功能由用户命令 ipvsadm 和 IPVS 内核模块中与其相应的部分共同完成，IPVS 内核的这部分代码主要在 ip_vs_ctl.c 中[27]。

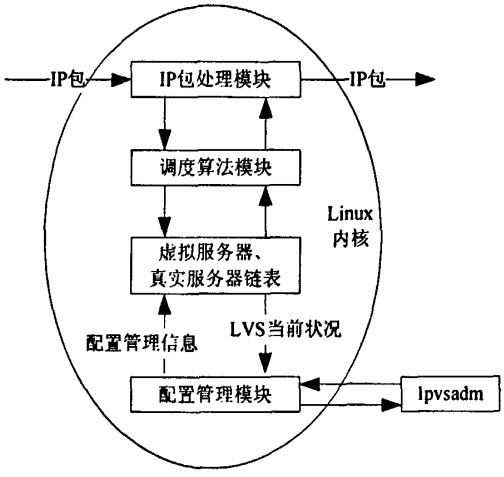

图 2-7 IPVS 结构

Figure2-7 Architecture of IPVS

2.5.1 IP 包处理

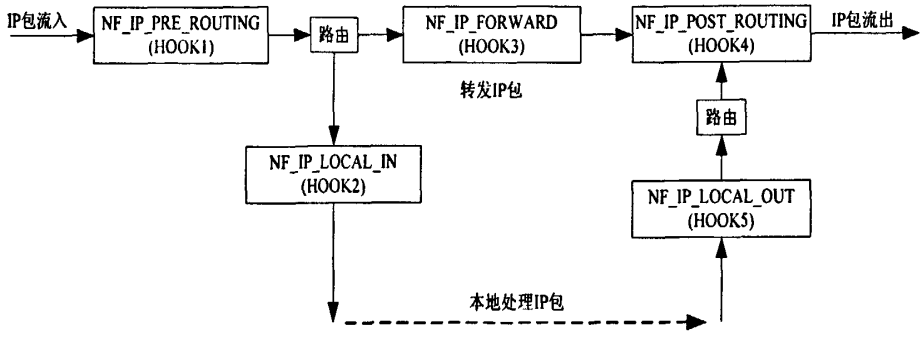

图 2-8　IP 报文在 Netfilter 中的数据流程

Figure2-8 Flow Chart of IP Packet in Netfilter

IP 包处理用 Linux 内核的 Netfilter 框架完成。一个数据包通过 Netfilter 框架的过程如图 2-8。

通俗的说，Netfilter 的架构就是在整个网络流程的若干位置放置了一些检测点(HOOK)，而在每个检测点上上登记了一些处理函数进行处理(如包过滤，NAT 等，甚至可以是用户自定义的功能)[28]

从图中可以看到 Netfilter 一共有 5 个钩子函数，分别为：

(1) NF_IP_PRE_ROUTING [HOOK1]：刚刚进入网络层的数据包通过此点(刚刚进行完版本号，校验和等检测)，源地址转换在此点进行；

(2) NF_IP_LOCAL IN[HOOK2]：经路由查找后，送往本机的通过此检查点，INPUT 包过滤在此点进行；

(3) NF_IP_FORWARD [HOOK3]：要转发的包通过此检测点，FORWORD 包过滤在此点进行；

(4) NF_IP_POST_ROUTING[HOOK4]：所有马上便要通过网络设备出去的包通过此检测点，内置的目的地址转换功能(包括地址伪装)在此点进行；

(5) NF_IP_LOCAL OUT[HOOK5]：本机进程发出的包通过此检测点，OUTPUT 包过滤在此点进行。

2.5.2 LVS 中 Netfilter 的实现

利用 Netfilter，LVS 处理数据报从左边进入系统，进行 IP 校验以后，数据报经过第一个钩子函数 NF_IP_PRE_ROUTING [HOOK1]进行处理；然后进行路由选择，决定该数据报是需要转发还是发给本机；若该数据报是发被本机的，则该数据经过钩子函数 NF_IP_LOCAL_IN[HOOK2]处理后传递给上层协议；若该数据报应该被转发，则它被 NF_IP_FORWARD[HOOK3]处理；经过转发的数据报经过最后一个钩子函数 NF_IP_POST_ROUTING[HOOK4]处理以后，再传输到网络上。本地产生的数据经过钩子函数 NF_IP_LOCAL_OUT[HOOK5]处理后，进行路由选择处理，然后经过 NF_IP_POST_ROUTING[HOOK4]处理后发送到网络上。

当启动 IPVS 加载 ip_vs 模块时，模块的初始化函数 ip_vs_init( )注册了 NF_IP_LOCAL_IN[HOOK2],NF_IP_FORWARD[HOOK3],NF_IP_POST_ROUTING[HOOK4] 钩子函数用于处理进出的数据报。

(1) NF_IP_LOCAL_IN 处理过程

用户向虚拟服务器发起请求，数据报经过 NF_IP_LOCAL_IN[HOOK2],进入 ip_vs_in( )进行处理。如果传入的是 icmp 数据报，则调用 ip_vs_in_icmp( )；否则继续判断是否为 tcp/udp 数据报，如果不是 tcp/udp 数据报，则函数返回

NF_ACCEPT(让内核继续处理该数据报)；余下情况便是处理 tcp/udp 数据报。

首先，调用 ip_vs_header_check( )检查报头，如果异常，则函数返回 NF_DROP(丢弃该数据报)。接着，调用 ip_vs_conn_in_get( )去 ip_vs_conn_tab 表中查找是否存在这样的连接：它的客户机和虚拟服务器的 ip 地址和端口号以及协议类型均与数据报中的相应信息一致。如果不存在相应连接，则意味着连接尚未建立，此时如果数据报为 tcp 的 sync 报文或 udp 数据报则查找相应的虚拟服务器；如果相应虚拟服务器存在但是已经满负荷，则返回 NF_DROP；如果相应虚拟服务器存在并且未满负荷，那么调用 ip_vs_schedule( )调度一个 RS 并创建一个新的连接，如果调度失败则调用 ip_vs_leave( )继续传递或者丢弃数据报。如果存在相应连接，首先判断连接上的 RS 是否可用，如果不可用则处理相关信息后返回 NF_DROP。找到已存在的连接或建立新的连接后，修改系统记录的相关信息如传入的数据报的个数等。如果这个连接在创建时绑定了特定的数据报传输函数，调用这个函数传输数据报，否则返回 NF_ACCEPT。

ip_vs_in()调用的 ip_vs_in_icmp( )处理 icmp 报文。函数开始时检查数据报的长度，如果异常则返回 NF_DROP。函数只处理由 tcp/udp 报文传送错误引起的目的不可达、源端被关闭或超时的 icmp 报文，其他情况则让内核处理。针对上述三类报文，首先检查检验和。如果检验和错误，直接返回 NF_DROP；否则，分析返回的 icmp 差错信息，查找相应的连接是否存在。如果连接不存在，返回 NF_ACCEPT；如果连接存在，根据连接信息，依次修改差错信息包头的 ip 地址与端口号及 ICMP 数据报包头的 ip 地址，并重新计算和修改各个包头中的检验和，之后查找路由调用 ip_send( )发送修改过的数据报，并返回 NF_STOLEN(退出数据报的处理过程)。

ip_vs_in()调用的函数 ip_vs_schedule( )为虚拟服务器调度可用的 RS 并建立相应连接。它将根据虚拟服务器绑定的调度算法分配一个 RS，如果成功，则调用 ip_vs_conn_new( )建立连接。ip_vs_conn_new( )将进行一系列初始化操作：设置连接的协议、ip 地址、端口号、协议超时信息，绑定 application helper、RS 和数据报传输函数，最后调用 ip_vs_conn_hash( )将这个连接插入哈希表 ip_vs_conn_tab 中。一个连接绑定的数据报传输函数，依据 IPVS 工作方式可分为 ip_vs_nat_xmit( ),ip_vs_tunnel_xmit( ),ip_vs_dr_xmit( )。例如 ip_vs_nat_xmit( )的主要操作是：修改报文的目的地址和目的端口为 RS 信息，重新计算并设置检验和，调用 ip_send( )发送修改后的数据报。

(2) NF_IP_FORWARD 处理过程

数据报进入 NF_IP_FORWARD 后，将进入 ip_vs_out( )进行处理。这个函数只在 NAT 方式下被调用。它首先判断数据报类型，如果为 icmp 数据报则直接调用 ip_vs_out_icmp( )；其次判断是否为 tcp/udp 数据报，如果不是这二者则返回

NF_ACCEPT。余下就是 tcp/udp 数据报的处理。首先，调用 ip_vs_header_check( ) 检查报头，如果异常则返回 NF_DROP。其次，调用 ip_vs_conn_out_get( )判断是否存在相应的连接。若不存在相应连接：调用 ip_vs_lookup_real_service( )去哈希表中查找发送数据报的 RS 是否仍然存在，如果 RS 存在且报文是 tcp 非复位报文或 udp 报文，则调用 icmp_send( )给 RS 发送目的不可达 icmp 报文并返回 NF_STOLEN；其余情况下均返回 NF_ACCEPT。若存在相应连接：检查数据报的检验和，如果错误则返回 NF_DROP，如果正确，修改数据报，将源地址修改为虚拟服务器 ip 地址，源端口修改为虚拟服务器端口号，重新计算并设置检验和，并返回 NF_ACCEPT。

ip_vs_out_icmp( )的流程与 ip_vs_in_icmp( )类似，只是修改数据报时有所区别：ip 报头的源地址和差错信息中 udp 或 tcp 报头的目的地址均修改为虚拟服务器地址，差错信息中 udp 或 tcp 报头的目的端口号修改为虚拟服务器的端口号。

(3) NF_IP_POST_ROUTING 处理过程

NF_IP_POST_ROUTING 钩子函数只在 NAT 方式下使用。数据报进入 NF_IP_POST_ROUTING 后,由 ip_vs_post_routing( )进行处理。它首先判断数据报是否经过 IPVS，如果未经过则返回 NF_ACCEPT；否则立刻传输数据报，函数返回 NF_STOLEN，防止数据报被 iptable 的规则修改。

2.5.3 LVS 系统配置与管理

IPVS 模块初始化时注册了 setsockopt/getsockopt( )，ipvsadm 命令调用这两个函数向 IPVS 内核模块传递 ip_vs_rule_user 结构的系统配置数据，完成系统的配置，实现虚拟服务器和 RS 地址的添加、修改、删除操作。系统通过这些操作完成对虚拟服务器和 RS 链表的管理。

虚拟服务器的添加操作由 ip_vs_add_service( )完成，该函数根据哈希算法向虚拟服务器哈希表添加一个新的节点，查找用户设定的调度算法并将此算法绑定到该节点；虚拟服务器的修改由 ip_vs_edit_service( )完成，此函数修改指定服务器的调度算法；虚拟服务器的删除由 ip_vs_del_service( )完成，在删除一个虚拟服务器之前，必须先删除此虚拟服务器所带的所有 RS，并解除虚拟服务器所绑定的调度算法。

与之类似，RS 的添加、修改、删除操作分别由 ip_vs_add_dest( )、ip_vs_edit_dest( )和 ip_vs_edit_server( )完成。

2.6 本章小结

本章主要描述了Linux Virtual Server集群系统。首先介绍了LVS集群的概念、体系结构及服务框架，然后详细分析了 LVS 集群中的三种负载均衡技术并对三者进行了比较分析。后面几章的内容都是基于本章的研究结果。

第 3 章 动态反馈负载均衡算法的设计和实现

LVS 集群的任务调度策略的关键就在于系统内核中的负载均衡调度算法的效率，针对不同的网络服务需求和服务器配置，LVS 负载均衡调度实现了八种负载调度算法，用户可以根据应用情况的不同选择最佳算法组合，这些算法都是在 LVS 体系结构中的负载调度器(Load Balancer)上实现。其主要的功能就是将客户的请求均衡地分配到各服务器上。

本章将讨论负载调度器上的负载均衡调度算法。首先给出了系统在内核中所实现的各种连接调度算法；然后在原有算法基础上提出了一个动态反馈负载均衡算法(Dynamic Feedback Load Balancing)，它结合内核中的加权连接调度算法，根据动态反馈回来的负载信息调整服务器的权值，来进一步避免服务器间的负载倾斜。

3.1 LVS 内核中的调度算法

LVS 在内核中的负载均衡调度是以连接为粒度的。在 HTTP 协议（非持久）中，每个对象从 WEB 服务器上获取都需要建立一个 TCP 连接，同一用户的不同请求会被调度到不同的服务器上，所以这种细粒度的调度在一定程度上可以避免单个用户访问的突发性引起服务器间的负载不平衡。

在内核中的连接调度算法上，IPVS 已实现了以下八种调度算法[21][26]：

- 轮叫调度（Round-Robin Scheduling）
- 加权轮叫调度（Weighted Round-Robin Scheduling）
- 最小连接调度（Least-Connection Scheduling）
- 加权最小连接调度（Weighted Least-Connection Scheduling）
- 基于局部性的最少链接（Locality-Based Least Connections Scheduling）
- 带复制的基于局部性最少链接（Locality-Based Least Connections with Replication Scheduling）
- 目标地址散列调度（Destination Hashing Scheduling）
- 源地址散列调度（Source Hashing Scheduling）

这八种算法中前面四种最常用，所以重点给出前面四种连接调度算法的实现过程。

3.1.1 轮转调度(RR)

轮叫调度(RR)算法就是以轮叫的方式依次将请求调度到不同的服务器，即每次调度执行 i = (i + 1) mod n，并选出第 i 台服务器。算法的优点是其简洁性，它无需记录当前所有连接的状态，所以它是一种无状态调度。

算法实现过程：为了实现轮询，LVS 按照服务的类型，把真实服务器连接成支持不同服务类型的双向循环链表，对于某种类型的服务来说，算法首先找到支持该服务类型的真实服务器链表，然后在此链表中从头部开始查询，如果找到目前未被连接的、而且服务器的权值大于零的真实服务器，则返回真实服务器的指针，否则表示当前没有可用的真实服务器，返回为空指针。

在系统实现时，引入了一个额外条件，当服务器的权值为零时，表示该服务器不可用而不被调度。这样做的目的是将服务器切出服务（如屏蔽服务器故障和系统维护），同时与其他加权算法保持一致。

它的算法流程如图 3-1。假设有 n 个服务器 $S=\{S_0, S_1, \cdots, S_{n-1}\}$，指示变量 i 表示上一次选择的服务器，指示变量 j 表示真实服务器链表中第 j 个服务器，W(Si)表示服务器 Si 的权值。变量 i 被初始化为 n-1，其中 n>0。

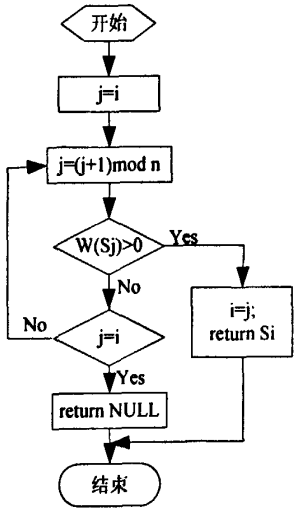

图 3-1 轮转调度算法流程图

Figure3-1 Flow Chart of Round Robin Scheduling

轮叫调度算法假设所有服务器处理性能均相同，不管服务器的当前连接数和响应速度。该算法相对简单，不适用于服务器组中处理性能不一的情况，而且当请求服务时间变化比较大时，轮叫调度算法容易导致服务器间的负载不平衡。

虽然 Round Robin DNS 方法也是以轮叫调度的方式将一个域名解析到多个 IP 地址，但轮叫 DNS 方法的调度粒度是基于每个域名服务器的，域名服务器对域名解析的缓存会妨碍轮叫解析域名生效，这会导致服务器间负载的严重不平

衡。这里，IPVS 轮叫调度算法的粒度是基于每个连接的，同一用户的不同连接都会被调度到不同的服务器上，所以这种细粒度的轮叫调度要比 DNS 的轮叫调度优越很多。

3.1.2 加权轮转调度(WRR)

加权轮叫调度算法可以解决服务器间性能不一的情况，它用相应的权值表示服务器的处理性能，服务器的缺省权值为 1。假设服务器 A 的权值为 1，B 的权值为 2，则表示服务器 B 的处理性能是 A 的两倍。加权轮叫调度算法是按权值的高低和轮叫方式分配请求到各服务器。权值高的服务器先收到的连接，权值高的服务器比权值低的服务器处理更多的连接，相同权值的服务器处理相同数目的连接数。

算法实现过程：在 LVS 中，按照服务类型，把真实服务器连接成支持不同服务类型的循环链表，变量 cl 指向上次分配的真实服务器，变量 cw 记录了上次分配的真实服务器的权值。当负载均衡器分配用户请求时，算法从 cl 开始搜索循环链表，当找到一个权值大于或等于当前权值 cw 时，就停止搜索，同时将 cl 改为指向此结点并返回指向此结点的指针值。如果没有找到符合要求的服务器，则查找此时权值最大的服务器；如果此时权值最大的服务器的权值大于零，则返回此服务器的指针；否则返回空指针。

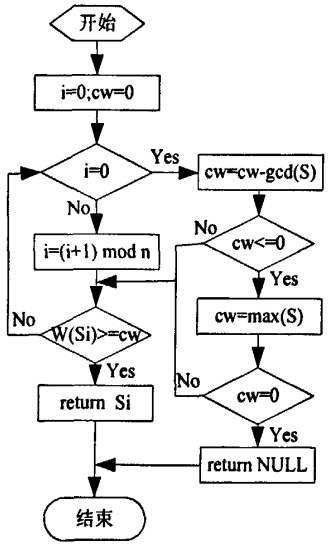

图 3-2 加权轮转调度算法流程图

Figure3-2 Flow Chart of Weighted Round Robin Scheduling

它的算法流程如图 3-2。假设有 n 个服务器 S={S_0，S_1，…，S_{n-1}}，W(Si)表示服务器 Si 的权值，一个指示变量 i 表示上一次选择的服务器，指示变量 cw 表示当前调度的权值，max(S)表示集合 S 中所有服务器的最大权值，gcd(S)表示

集合 S 中所有服务器权值的最大公约数。变量 i 和 cw 最初都被初始化为零。

从上面的算法流程中，可以看出当服务器的权值为零时，该服务器不被被调度;当所有服务器的权值为零，即对于任意 i 有 W(si)=0，则没有任何服务器可用，算法返回 NULL，所有的新连接都会被丢掉。加权轮叫调度也无需记录当前所有连接的状态，所以它也是一种无状态调度。

3.1.3 最小连接调度(LC)

最小连接调度算法是把新的连接请求分配到当前连接数最小的服务器。最小连接调度是一种动态调度算法，它通过服务器当前所活跃的连接数来估计服务器的负载情况。调度器需要记录各个服务器已建立连接的数目，当一个请求被调度到某台服务器，其连接数加 1；当连接中止或超时，其连接数减 1。

算法的实现：在 LVS 中，把连接进一步分为活动连接和静止连接。静止连接就是指刚建立的处于监听状态的连接；动态连接就是指开始接收或发送数据的连接。一般说来服务器处理活动连接所需要消耗的资源比处于静止连接所需消耗的资源要大得多，所以真实服务器负载的真正连接数为：活动连接数×256+静止连接数。这里的比例系数 256 是通过实验估计的，对不同的系统环境，此系数可能有所不同。当负载均衡器分配服务时，会在支持此服务的真实服务服务器分配链表中，找到一个连接数最小、且权值大于零的服务器，并返回此服务器的指针，否则返回空指针。

它的算法流程如图 3-3。假设有 n 个服务器 $S=\{S_0, S_1, \cdots, S_{n-1}\}$，W(Si)表示服务器 Si 的权值，C(Si)表示服务器 Si 的当前连接数，指示变量 m 表示真实服务器链表中第 m 个服务器。

当各个服务器有相同的处理性能时，最小连接调度算法能把负载变化大的请求平滑分布到各个服务器上，所有处理时间比较长的请求不可能被发送到同一台服务器上。但是，当各个服务器的处理能力不同时，该算法并不理想，因为 TCP 连接处理请求后会进入 TIME_WAIT 状态，TCP 的 TIME_WAIT 一般为 2 分钟，此时连接还占用服务器的资源，所以会出现这样情形，性能高的服务器已处理所收到的连接，连接处于 TIME_WAIT 状态，而性能低的服务器已经忙于处理所收到的连接，还不断地收到新的连接请求。

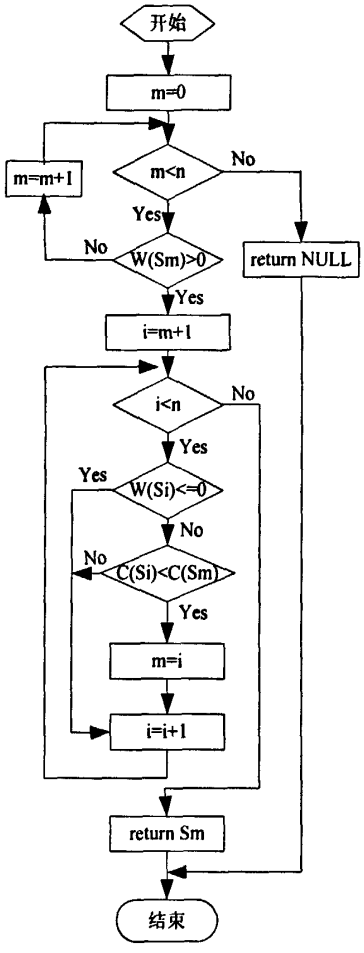

图 3-3 最小连接调度算法流程图

Figure3-3 Flow Chart of Least Connection Scheduling

3.1.4 加权最小连接调度(WLC)

加权最小连接调度算法是最小连接调度的超集，各个服务器用相应的权值表示其处理性能。服务器的缺省权值为 1，系统管理员可以动态地设置服务器的权值。加权最小连接调度在调度新连接时尽可能使服务器的已建立连接数和其权值成比例。

算法的实现：当负载均衡器分配服务时，会在支持此服务的真实服务器分配链表中，从头部开始寻找连接数与权值的比值最小、且权值不为零(表示服务器可用)的服务器，如果找到符合要求的服务器，则返回此服务器指针，否则返回空指针。

它的算法流程如图 3-4。假设有 n 个服务器 $S=\{S_0, S_1, \cdots, S_{n-1}\}$，W(Si)表示服务器 Si 的权值，C(Si)表示服务器 Si 的当前连接数，指示变量 m 表示真实服务器链表中第 m 个服务器。所有服务器当前连接数的总和为 CSUM

$=\sum C(Si)\ (i=0, 1, .., n-1)$。当前的新连接请求会被发送服务器 Sm，当且仅当服务器 Sm 满足以下条件

$$(C(Sm) / CSUM) / W(Sm) = \min \{ (C(Si) / CSUM) / W(Si) \} \quad (i=0, 1, ., n-1)$$

其中 W(Si)不为零，因为 CSUM 在这一轮查找中是个常数，所以判断条件可以简化为

$$C(Sm) / W(Sm) = \min \{ C(Si) / W(Si) \} \quad (i=0, 1, ., n-1)$$

其中 W(Si)不为零因为除法所需的 CPU 周期比乘法多，且在 Linux 内核中不允许浮点除法，服务器的权值都大于零，所以判断条件 C(Sm) / W(Sm) > C(Si) / W(Si) 可以进一步优化为 C(Sm)*W(Si) > C(Si)* W(Sm)。同时保证服务器的权值为零时，服务器不被调度。

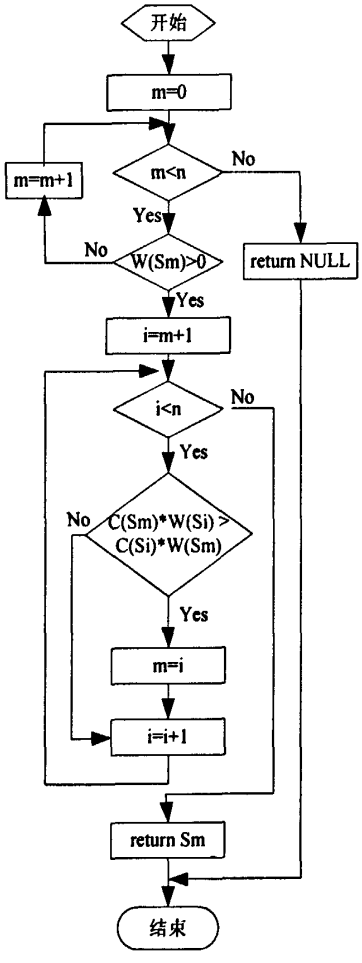

图 3-4 加权最小连接调度算法流程图

Figure3-4 Flow Chart of Weighted Least Connection Scheduling

算法既考虑了集群中各真实服务器的处理性能的不同，又考虑了各个真实服务器的状态，可以实现真正意义上的负载均衡。虽然算法比前面所述 3 种算法的复杂度稍微大些，但是它提高了系统的效率，这种效率的提高不仅抵消了算法

所带来的开销，而且大大地改善了系统地性能。WLC 是 LVS 的缺省的负载分配算法，在通常的应用中一般都采用此算法。

3.1.5 其它算法

除了上述四种简单常用的调度算法外，还有另外四种调度算法：

(1)基于局部性的最少链接调度(LBLC)算法是针对请求报文的目标 IP 地址的负载均衡调度，目前主要用于 Cache 集群系统，因为在 Cache 集群中客户请求报文的目标 IP 地址是变化的。这里假设任何后端服务器都可以处理任一请求，算法的设计目标是在服务器的负载基本平衡情况下，将相同目标 IP 地址的请求调度到同一台服务器，来提高各台服务器的访问局部性和主存 Cache 命中率，从而整个集群系统的处理能力。

LBLC 调度算法先根据请求的目标 IP 地址找出该目标 IP 地址最近使用的服务器，若该服务器是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于其一半的工作负载，则用“最少链接”的原则选出一个可用的服务器，将请求发送到该服务器。

(2)带复制的基于局部性最少链接调度(LBLCR)算法也是针对目标 IP 地址的负载均衡，目前主要用于 cache 集群系统。它与 LBLC 算法的不同之处是它要维护从一个目标 IP 地址到一组服务器的映射，而 LBLC 算法维护从一个目标 IP 地址到一台服务器的映射。对于一个“热门”站点的服务请求，一台 Cache 服务器可能会忙不过来处理这些请求。这时，LBLC 调度算法会从所有的 Cache 服务器中按“最少连接”原则选出一台 Cache 服务器，映射该“热门”站点到这台 Cache 服务器，很快这台 Cache 服务器也会超载，就会重复上述过程选出新的 Cache 服务器。这样，可能会导致该“热门”站点的映像会出现在所有的 Cache 服务器上，降低了 Cache 服务器的使用效率。LBLCR 调度算法将“热门”站点映射到一组 Cache 服务器(服务器集合)，当该“热门”站点的请求负载增加时，会增加集合里的 Cache 服务器，来处理不断增长的负载；当该“热门”站点的请求负载降低时，会减少集合里的 Cache 服务器数目。这样，该“热门”站点的映像不太可能出现在所有的 Cache 服务器上，从而提高 Cache 集群系统的使用效率。

LBLCR 算法先根据请求的目标 IP 地址找出该目标 IP 地址对应的服务器组；按“最小连接”原则从该服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器;若服务器超载，则按“最小连接”原则从整个集群中选出一台服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。

(3)目标地址散列调度(DH)算法也是针对目标 IP 地址的负载均衡，但它是一种静态映射算法，通过一个散列(Hash)函数将一个目标 IP 地址映射到一台服务器。目标地址散列调度算法先根据请求的目标 IP 地址，作为散列键(HashKey)从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

(4)源地址散列调度(SH)算法正好与目标地址散列调度算法相反，它根据请求的源 IP 地址，作为散列键(HashKcy)从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。它采用的散列函数与目标地址散列调度算法的相同。它的算法流程与目标地址散列调度算法的基本相似，除了将请求的目标 IP 地址换成请求的源 IP 地址，所以这里不一一叙述。

在实际应用中，源地址散列调度和目标地址散列调度可以结合使用在防火墙集群中，它们可以保证整个系统的唯一出入口。

通过对以上 8 种常用调度算法的分析可以发现，可以分为两大类：一类静态负载均衡算法，它不考虑服务器工作时的负载情况，而只是根据事先定好的策进行任务请求的分配；另一类是动态负载均衡算法，它以系统的当前负载情况分配依据，考虑到服务器的处理能力，充分利用系统资源，从而减小服务的响应时间。因此动态负载均衡算法要比静态负载均衡算法更有实用性。

在实际应用中，加权最小连接算法(WLC)是最常用的，也是负载均衡效果较好的一个算法。WLC 算法考虑了各个服务节点的实际性能的差异，以权重表示服务节点的处理能力，从而在各个服务几点之间合理的分配负载。但是仍然存在着以下几个不足的地方[29]：

(1)它以连接数来表示节点负载，不能较准确的反映服务节点的负载情况。

在 WWW 服务中，客户连接请求的服务对服务器资源的要求差异很大，如服务时间、流量、CPU 资源等，因此仅仅依靠连接数来衡量负载是不够的。

(2)不能动态反映服务器的负载情况。网络服务的服务时间、访问分布和所需资源存在不均匀性，而上述各调度算法中服务器的权值是事先设置好的，不能动态反映服务器的负载情况，无法实现系统内服务器之间的真正均衡。

因此有必要改进上述算法，使负载均衡器能够较准确的了解各个服务节点的负载状况，然后根据负载的情况选取最合适的服务节点，从而实现以下目标[30]：

(l)保证系统在长时间运行状态下个节点的负载不发生较大的倾斜；

(2)充分利用节点的处理能力；

(3)在保证前两个目标的前提下，尽量减小算法的复杂度和系统开销。

3.2 动态反馈负载均衡算法设计

3.2.1 动态反馈负载均衡算法的提出

当客户通过 TCP 连接访问网络时，服务所需时间和所要消耗的计算资源是千差万别的，它依赖于很多因素。例如，它依赖于请求的类型、当前网络带宽和当前服务器资源利用情况等。一些负载比较重的请求需要进行计算密集的查询或具有很长的应答数据流，而负载比较轻的请求往往只需要读取一个 HTML 页面或者进行很简单的计算。因此，仅仅通过连接数并不能准确的反映服务器的真实负载，这就需要通过结合其他的参数来估计服务器的负载状况，通常的参数有 CPU 利用率、磁盘使用情况、内存利用情况、当前的进程数和服务器提供服务的响应时间。在上述参数中，CPU 利用率体现了服务器的处理能力，而服务器响应时间能比较好的反映服务器上的请求等待队列的长度。

通过研究发现，以 LVS 的八种调度算法为基础，现在人们也研究出很多种动态调度算法，如动态负反馈[31-32]、基于 Agent 的调度算法[33]、最快回应最少连接调度算法[34]、最低利用率最少连接算法[35]等。这些动态调度算法的基本思想都是负载均衡器周期性地从后端服务器获取当前各节点真实服务器的负载状态，并根据这些状态来动态调整后续的分配策略，其中最为关键的步骤是动态地计算每台服务器的当前负载权值。以真实服务器的实时负载为依据，动态计算并调整服务器权值，体现了服务器的真实处理能力，克服了静态调度算法的缺点，更加科学合理。但是权值的调整是一个随着访问需求变化不断振荡的过程，也就是说动态权值计算的过程将一直进行下去，这无疑会对负载均衡器的资源造成浪费，降低系统的效率。

综合分析各种动态调度算法后，本文在加权最小连接调度算法的基础上，提出了一种改进的动态反馈负载均衡算法 DF[27-35]，通过负载均衡器向每台真实服务器发送请求，服务器将自身的 CPU 利用率作为回应返回给负载均衡器，负载均衡器根据请求响应时间和返回的 CPU 利用率以及分配的连接数来确定服务器的当前负载情况，从而将新到的请求发送到负载最小的服务器上。DF 算法以连接数、CPU 利用率和服务响应时间的综合计算值作为服务器的真实负载，并据此进行连接的调度。这样就能够根据服务器的真实负载状况进行连接的动态调度。

3.2.2 动态反馈负载均衡算法基本思想

和加权最小连接数算法相比，改进的动态反馈负载均衡算法 DF 将影响服务

器负载的 CPU 利用率和响应时间这两个参数考虑在内，通过负载均衡器定时的给服务器发送一个特殊请求，而服务器将自身的 CPU 利用率作为回应返回给负载均衡器，负载均衡器根据返回的 CPU 利用率及响应时间计算各服务器的负载情况。当然，负载均衡器向服务器发送特殊请求的周期不能太短，因为过于频繁的采集服务器节点的负载信息不仅会增加网络开销，而且对负载均衡器以及服务器的资源也是一种浪费。具体实现时适当的调整均衡器采集服务器节点负载信息的周期，这个时间间隔一般设置为 5~20 秒。

改进的动态反馈负载均衡算法 DF 的流程如图 3-5：

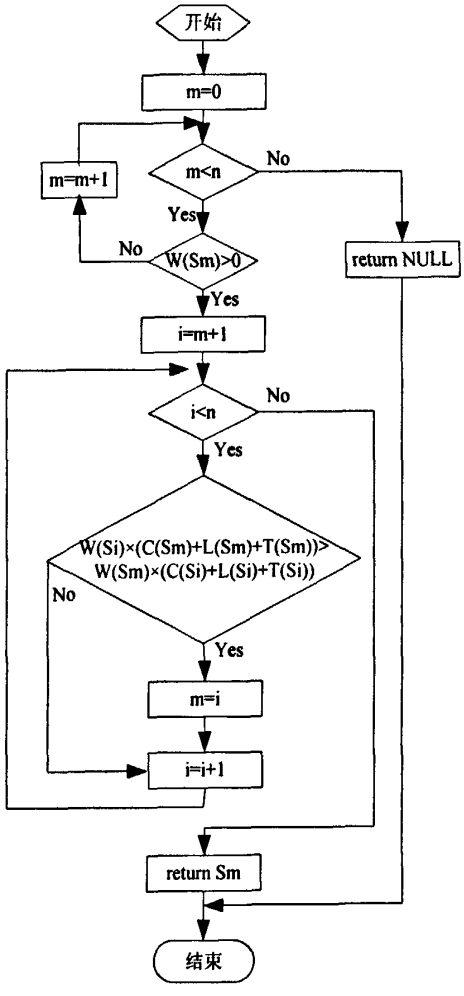

图 3-5 动态反馈负载均衡算法 DF 流程图

Figure3-5 Flow Chart of Dynamic Feedback Scheduling

假设有 n 个服务器 $S=\{S_0, S_1, \cdots, S_{n-1}\}$，W(Si)表示服务器 Si 的权值，C(Si)表示服务器 Si 的当前连接数，L(Si)表示服务器 Si 的 CPU 利用率，T(Si)表示服务器 Si 响应负载均衡器发出的特殊请求的响应时间，指示变量 m 表示真实服务器链表中第 m 个服务器。当有新的连接请求到来时，它将被发送到服务器 Sm，

当且仅当服务器 Sm 满足以下条件：

W(Sm)×(C(Sm)+L(Sm)+T(Sm))=min{W(Si)×(C(Si)+L(Si)+T(Si))} (i=0, 1, ... , n-1)。

该算法不仅考虑了每个服务器节点的请求连接数，而且加入了反映服务器负载状况的 CPU 利用率和响应时间两个参数，通过反馈信息不断的调整任务分配，从而避免了有些节点在超载时仍然收到大量请求的情况，提高了整个系统的服务质量和吞吐率。

3.3 动态反馈负载均衡算法的实现

DF 算法是在 WLC 的基础上的改进，主要工作是在负载均衡器(Balancer)和真实服务器(Realserver)加入相应的模块，其它的工作仍然由 LVS 来完成。Realserver 端需要加入负载信息收集模块，Balancer 端需要加入负载信息查询模块 [36-48]。引入动态反馈调度算法后的模块关系如图 3-6：

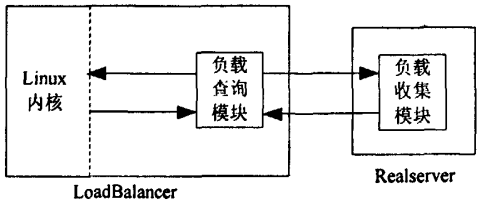

图 3-6 引入动态反馈算法后的模块关系图

Figure3-6 chart of module's relation with Dynamic Feedback algorithm

具体负载均衡器和真实服务器上的模块设计如下：

1 Balancer 模块

主要功能是从真实服务器上获取各个服务器的真实负载信息，并通过计算得出负载最小的服务器，从而将新到的请求转发至该服务器。

具体实现：Balancer 定时向每个真实服务器发送一个请求，然后接受服务器返回的反映负载状况的 CPU 利用率信息，并记录请求的响应时间，修改服务器节点的 CPU 及 rt 数据项，根据算法计算出 Load 值最小服务器，并将指针指向服务器链表中的对应服务器，准备接受并处理新到的请求。Balancer 模块流程如图 3-7。

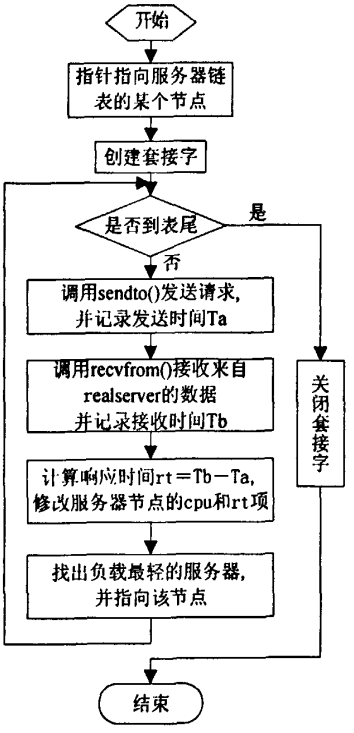

图 3-7 Balancer 模块流程图

Figure3-7 flow chart of Balancer module

2 Realserver 模块

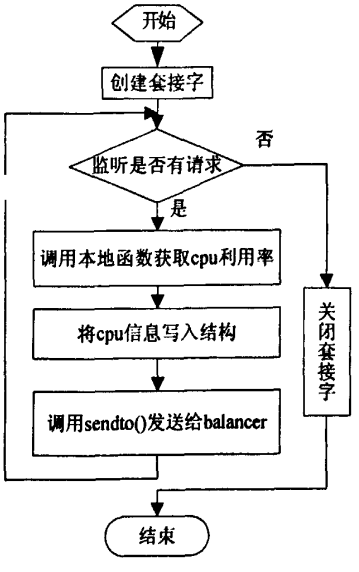

图 3-8 Realserver 模块流程图

Figure3-8 flow chart of Realserver module

主要功能是当收到来自 Balancer 的请求后，采集本机 CPU 的利用率，并发送给 Balancer。

具体实现：Realsevrer 一直在监听是否有来自 Balancer 的请求，如果有，则

运行采集本机 CPU 利用率的函数，并将结果发送给 Balancer。Realsevrer 模块流程图如图 3-8。

3 主要数据结构

由于改进算法在计算负载时不仅考虑了服务器的当前连接数，还涉及到服务器的 CPU 利用率及响应时间，因此对于服务器的原有数据结构有必要进行一定的修改，具体修改是在原有的服务器数据结构中添加两项分别用来表示服务器的 CPU 利用率及响应时间[49]。

```
struct ip_vs_dest{
        ……
        atomic_t   cpu;   /*cpu 利用率*/
        atomic_t   rt;   /*响应时间*/
        ……
}
```

4 动态反馈调度模块 df

在具体实现时，df 模块是一个可加载模块[50]。核心实现函数是 ip_vs_df_schedule()，当负载均衡器分配任务时，将调用此函数找到一个负载最轻的服务器，并返回该服务器的指针，否则返回空指针。

ip_vs_df_schedule()函数的类代码如下：

```
static struct ip_vs_dest *
ip_vs_df_schedule(struct ip_vs_service *svc, const struct sk_buff *skb)
{
   从头开始遍历真实服务器链表{      /*目的是找出第 1 个可用的服务器*/
      IF (当前服务器 dest 不过载 AND dest 的权值 weight>0){
         least = dest;
         获得当前服务器 dest 的连接数 loh;
         goto nextstage;
      }
      ELSE
         return NULL;
   }
   nextstage:
   从 least 的下一个服务器开始遍历真实服务器链表{
   /*目的是找出负载最轻的服务器*/
      IF 当前服务器 dest 过载 then 跳过 dest 继续遍历;
      获得当前服务器 dest 的连接数 doh;
```

```
        IF(W(dest)*(loh+T(least)+L(least))> W(least)*(doh+T(dest)+L(dest))){
        /*W(Si)为服务器 Si 的权值, T(Si)为服务器 Si 对来自负载均衡器请求的响应时间, L(Si)为服务器 Si 的 CPU 利用率*/
        least=dest;
        loh=doh;
        }
    }
return least;                    /*least 即为当前负载最轻的服务器*/
}
```

5 CPU 利用率的获取

当真实服务器在特定特定端口监测到有来自负载均衡器的请求，就运行获取 CPU 利用率的函数 cpu()。真实服务器上运行的是 Linux 操作系统，Linux 内核提供了一种通过/proc 文件系统，可以被用于收集有用的关于系统和运行中的内核的信息。/proc 文件系统是一个虚拟的文件系统，它通过文件系统的接口实现，用于输出系统运行状态。它以文件系统的形式，为操作系统本身和应用进程之间的通信提供了一个界面，使应用程序能够安全、方便地获得系统当前的运行状况何内核的内部数据信息，并可以修改某些系统的配置信息[51-52]。由于/proc 以文件系统的接口实现，因此可以象访问普通文件一样访问它，获取 CPU 相关信息可以通过下述语句实现：fp1=openfile("/proc/stat","r")，该语句可以获取 CPU 的 user、nice、system、idle 信息，它们分别是 CPU 用户时间、CPU nice 时间、CPU 系统时间和 CPU 空闲时间，上述所有时间的和记做 total，表示 CPU 总的开销时间，CPU 利用率定义为 user、nice、system 之和与 total 的比值。

6 UDP 套接字通讯

为了实现负载均衡器与真实服务器之间的通讯，这里我们采用 UDP 套接字方式[52]，负载均衡器会每隔一段时间以客户端的身份在特定端口利用 socket 向真实服务器发送请求，服务器在相应端口监听到请求后就会收集本地 CPU 利用率，然后以同样的方式将数据发送给均衡器，从而完成了服务器与均衡器之间的通讯。

3.4 本章小结

本章首先分析了现有的八种 LVS 负载均衡算法，提出了这些算法的不足，针对这些不足之处，提出了一种改进的负载均衡算法，并描述了算法的基本思想。对于算法的实现，给出了服务器方及均衡器方的实现流程，并对实现过程中的一些关键问题进行了阐述。

第 4 章 WEB 服务器集群的组建与测试

4.1 系统设计目标

Web 服务器集群系统的设计目标有两个：一是提高网络教学系统服务器的吞吐率，解决访问瓶颈问题；二是提高服务器集群的可伸缩性、可用性和可管理性。

第 1 个目标可以通过 LVS 的 IP 负载均衡技术来解决，综合考虑 LVS 三种负载均衡模式，虽然 IP tunneling 和 Direct Routing 的工作效率要高于 NAT 模式，但是 IP tunneling 模式要求真实服务器的操作系统必须支持 IP 封装技术，也就是只能支持 Linux 操作系统；Direct Routing 模式虽然对真实服务器的操作系统没有要求，但是它要求每个真实服务器都必须具有自己的合法 IP 地址，因此也不太适合本文的环境，所以选择使用 NAT 工作模式，NAT 虽然效率低于前两者，但它配置容易、使用灵活、要求较低，在目前服务器数目不大的情况下完全可以满足系统的需求。

第 2 个目标可以通过 LVS 系统自带的管理软件 Ipvsadm 来实现，Ipvsadm 可以轻松的完成真实服务器的添加和删除以及服务器权值的修改等管理功能。通过与 Heartbeat 等软件包配合，可以实现对负载均衡器本身以及真实服务器健康状态进行监控[53]。

本文以校园网中的网络服务器为研究对象，利用 LVS 负载均衡技术设计一个具有可扩展、可提供 24×7 不间断服务、高可用等特性的集群系统。该系统以原有设备为设计基础，以现有主机及网络服务为设计对象，可以有效整合现有资源，提高资源利用率和系统性价比。

4.2 系统体系结构

Web 服务器集群系统的体系结构如图 4-1。

第 1 层负载均衡器。用户访问的唯一入口，也是集群系统的唯一入口点。它负责接收用户请求，并按一定均衡策略将用户请求分配给节点真实服务器。负载均衡器上安装节点服务器管理软件，可监控节点服务器意外宕机时切出服务器队列，或服务器检修、恢复功能后加入队列。考虑到设备的利用率，负载均衡器本身也作为一台真实服务器使用。

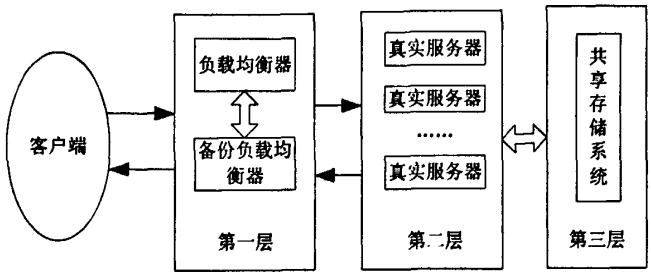

图 4-1 Web 服务器集群体系结构图

Figure4-1 Architecture of Web Server Cluster

第 2 层真实服务器池。实际处理客户端请求的网络服务器集合，一般可提供 WWW、FTP、Email 等网络服务。每台服务器又称为节点，处理来自均衡器分配的客户请求。服务器池可以是由配置相同的服务器组成，也可以是由配置不同的服务器组成，本系统是以原有设备为设计基础，以低成本条件下提高系统的吞吐量和可扩展性，因此设计时采用异构服务器组成的真实服务器池。

第 3 层共享存储系统。共享存储系统的作用是为真实服务器池提供一个共享存储区，以实现服务器池拥有相同的内容，提供相同的网络服务，从而保证真实服务池提供数据的一致性。根据存储数据类型、提供服务的不同，共享存储可以是数据库、网络文件系统和分布式文件系统，或是三都的混合。由于本系统提供的是以课件为主的静态文件，因此使用网络文件系统 NFS，在每个真实服务器的硬盘上划出一块区域作为共享存储。由于受设备限制，本层没有独立的存储设备，而是在第 2 层实现。

4.3 系统软硬件环境

4.3.1 硬件环境

考虑整个系统中，负载均衡器和真实服务器节点均需要硬件冗余，用以避免单一失效点问题。所以本系统设计时选用两台服务器用作负载均衡器，一台用做主负载均衡器，另一台用作备份负载均衡器，备份服务器同时也可作为后端的节点服务器；节点服务器至少配置两个节点，以后根据资金和业务情况再进行扩展；所有服务器最好采用支持 RAID，SCSI，热插拔等技术的专业服务器，以提供稳定高效的网络服务。具体硬件配置如下：

负载均衡器：CPU 为 P4 2.2GHz；内存 512MB；100M 网卡；操作系统为 Red Hat AS4.0，内核版本 2.6.26。

备份负载均衡器：(配置同负载均衡器)。

RS1 服务器：P4 2.0GHz；内存 512MB；100M 网卡；操作系统为 Red Hat AS4.0，内核版本 2.6.26。

RS2：(配置同 RS1 服务器)。

数据库服务器：无。

客户端：CPU 为 P4 2.2GHz；内存 512MB；100M 网卡；操作系统为 Windows XP。

4.3.2 软件环境

(1)Ipvsadm

Ipvsadm 是在 Linux 的内核中实现的，它在 Linux 的内核中监测需要路由的 IP 数据包，并根据用户设置的条件对数据包进行相应的操作。在 Linux 的内核中有三条控制数据包去向的链：input、forward、output，ipvsadm 是在 forward 的过程中对数据包进行操作的。ipvsadm 的作用是为用户选择合适的 web 服务器。

(2)Heartbeat

Hearbeat 是 Linux 高可用项目(又称为 Linux-HA)小组提供的开放、免费的源代码群集软件包，它使用心跳来监控群集中节点的“健康”状况。Heartbeat 通过通信介质(通常是串行设备和以太网)监控节点的“健康”状况。最好有多个冗余介质，以便我们既可以使用串行线又可以使用以太网链接。每个节点运行一个守护程序进程(称为“心跳”)。主守护程序派生出读和写每个心跳介质的子进程，以及状态进程。当检测到某个节点发生故障时，Heartbeat 运行 shell 脚本来启动(或停止)辅助节点上的服务。

(3)Ldirectord

Ldirectord(Linux Director Daemon)是 Jacob Rief 编程实现的一个独立进程，以实现对服务和物理服务器的监测，被广泛地用在 http 和 https 服务，能很好地与“heartbeat”配合工作。Ldirectord 主要是用来监视各个真实服务器的状态，专门用来和 LVS 调度器配合，使 ipvsadm 真正的完美起来的工具。当某一个真实服务器不能工作时，安装并设置好 Ldirectord 的调度器，可以很快得到消息，然后，把那台机器“踢”出服务器池，实际上是把它在调度器中的权值设成 0，让数据报不再发送给这个服务器。Ldirectord 可以方便地被 heartbeat 管理启动和停止，并能从 heartbeat 的配置文件中读取所有关于 IPVS 路由表配置的信息。VS/NAT 模式对后端服务器支持 Linux 操作系统或 Windows 操作系统；也可以是二者的混合。

(4)NFS

NFS 是一种分布式文件系统，其主要功能就是让网络上的 UNIX 主机可以共

享目录文件。它的原理是在客户端上，通过网络将远程主机共享的文件系统利用安装的方式加入本机的文件系统，此后的操作就像在本机操作一样。这样设计的好处是除了提升资源的使用率外，可以大大的节省硬盘的空间，因为每台主机不需要将所有文件都复制到本机上，同时也可做到资源的集中管理。

NFS 使用了一组活动的进程，为整个集群提供了一个统一存放 web 应用程序的目录。上传 web 应用程序到目录/usr/local/webs/后，在文件/etc/exports 中添加一行/usr/local/webs(rw)，这表明这个目录可以在装载 NFS 的任何远程系统上使用，并有读写权限。然后在后端的服务器上，在文件 etc/fstab 中将/usr/local/webs 映射为本地目录即可。

4.4 系统构建过程

4.4.1 负载均衡器上软件安装过程

(1) 安装 Ipvsadm。RedHat AS4.0 的内核版本为 2.6.26，与内核版本相对应的 Ipvsadm 的版本为 ipvsadm-1.24-6，其软件包可在网站 http://www.linuxvirtualserver.org/software/ipvs.html 下载。

进入目录/src/usr/ipvs-1.2.1/ipvs/ipvsadm，输入命令：

make

make install

然后输入 ipvsadm

若显示：Linux Virtual Server Version1.2.1(size=65536)

Prot Local Address：Port Scheduler Flags

~Remote Address：Port Forward Weight ActiveConn InActConn

则说明 ipvsadm 安装成功。

(2)将动态负载均衡算法 ip_vs_df_wlc.c 加入内核模块，并重新编译内核。

(3) 利用 ipvsadm 为集群配置服务和真实服务器，并将 RS1 和 RS2 的权值设置为 1，如下所示:

```
echo 0>/proc/sys/netlipv4/ip_ forward
ifconfig eth0:1 192.168.0.254/32 broadcast 192.168.0.254 up
route add -host 192.168.0.254 dev eth0:1
ipvsadm -C
ipvsadm -A -t 192.168.0.254:http -s df_wlc
ipvsadm -a -t 192.168.0.254:http -192.168.0.01 -g -w 1
ipvsadm -a -t 192.168.0.254:http -192.168.0.02 -g -w 1
```

4.4.2 真实服务器上软件安装过程

VS/NAT 模式负载均衡技术中，真实服务器节点上可以安装 linux 操作系统，或 Windows 操作系统。本文设计的系统中真实服务器节点上全部安装的是 Linux 操作系统，并安装 Apache 服务器提供 IIS 服务，接着再给各节点服务设置自己的 IP 地址和网关。LVS 集群采用静态调度算法时，Realsever 只要把自己的网关指向 Balancer 即可。当 LVS 集群采用动态反馈算法进行调度时，RS 上需运行客户端守护程序，一方面收集系统的状况；另一方面将负载信息处理后发送给 LB。

4.4.3 Ldirectord 的安装与配置

(1)下载高层 API 数库 libnet 软件包[54]和 heartbeat 软件包后，并按顺序先装 libnet 再安装 heartbeat 软件包包含的 ldirectord 工具。

命令格式：

./ConfigureMe configure disable swig disable smnp subagent

make

make install

rmp -ivh nodeps heartbeat-ldirectord-1.0.3.13 86.rpm

(2)配置 ldirectord.cf 文件。将/lusr/share/doc/heartbeat-ldirectord-1.0. 3/目录下 ldirectord 的配置文件范例 ldirectord.cf 文件，复制至/letc/ha. d/目录下，然后进行指定虚拟 IP 地址、真实服务器 IP 地址、fallback、服务类型指定和调度算法等内容的修改，并另存为 www.cf 或 mail.cf，以保证在 heartbeat 的配置文件 haresource 中使用。

4.4.4 Heartbeat 的安装与配置

(l)下载并安装 heartbeat 软件包。

命令格式：rmp -ivh nodeps heartbeat-1.0.3.i386.rpm

(2)heartbeat 配置

配置 heartbeat 时，需对三个文件进行修改：ha.cf, haresources, Authkeys 并把它们拷到/etclha. dl 目录下。

三个配置文件的作用及配置参数说明:

配置 Ha. cf

ha.cf 是 heartbeat 的主要配置文件[55]，可以对 heartbeat 的多数性能和状态进行配置。大部分选项的取值可以采用默认值，其中的主要选项及配置方法说明如

下：

debugfile /var/log/ha-debug：该文件保存 heartbeat 的调试信息

logfile /var/log/ha-log： heartbeat 的日志文件

keepalive 2：心跳的时间间隔，默认时间单位为秒

deadtime 30：超出该时间间隔未收到对方节点的心跳，则认为对方已经死亡。

warntime 10：超出该时间间隔未收到对方节点的心跳，则发出警告并记录到日志中。

initdead 120：在某些系统上，系统启动或重启之后需要经过一段时间网络才能正常工作，该选项用于解决这种情况产生的时间间隔。取值至少为 deadtime 的两倍。

udpport 694：设置广播通信使用的端口，694 为默认使用的端口号。

baud 19200：设置串行通信的波特率。

serial /dev/ttyS 0：选择串行通信设备，用于双机使用串口线连接的情况。如果双机使用以太网连接，则应该关闭该选项。

bcast eth0：设置广播通信所使用的网络接口卡。

Auto_failback on： heartbeat 的两台主机分别为主节点和从节点。主节点在正常情况下占用资源并运行所有的服务，遇到故障时把资源交给从节点并由从节点运行服务。在该选项设为 on 的情况下，一旦主节点恢复运行，则自动获取资源并取代从节点，否则不取代从节点。

ping ping-node 1 ping-node2：指定 ping node，ping node 并不构成双机节点，它们仅仅用来测试网络连接。

respawn hacluster /usr/lib/heartbeat/ipfail：指定与 heartbeat 一同启动和关闭的进程，该进程被自动监视，遇到故障则重新启动。最常用的进程是 ipfail，该进程用于检测和处理网络故障，需要配合 ping 语句指定的 ping node 来检测网络连接。

配置 haresources 文件：

haresources 文件用于指定双机系统的主节点、集群 IP、子网掩码、广播地址以及启动的服务等。其配置语句格式如下：

node-name network-config<resource-group>

其中 node-name 指定双机系统的主节点，取值必须匹配 ha.cf 文件中 node 选项设置的主机名中的一个，node 选项设置的另一个主机名成为从节点。

network-config 用于网络设置，包括指定集群 IP、子网掩码、广播地址等。

resource-group 用于设置 heartbeat 启动的服务，该服务最终由双机系统通过集群 IP 对外提供。

配置 authkeys 文件：

authkeys 文件用于 heartbeat 的鉴权设置，共有三种可用的鉴权方式：crc,md5 和 sha1。三种方式安全性依次提高，但同时占用的系统资源也依次扩大。crc 安全性最低，适用于物理上比较安全的网络，sha1 提供最为有效的鉴权方式，占用的系统资源也最多。

其配置语句格式如下：

auth <number>

<number> <authmethod> [<authkey>]

4.4.5 系统启动

在 HeartBeat 资源文件(/etc/ha.d/haresources)中定义了实现集群所需的各个软件的启动脚本。应将脚本拷贝至/letclinit.d 或/etc/ha.d/resource.d 目录下，启动顺序是：

loadbalancer LVS IPaddr:192.168.0.254/24/eth0 ipvsadm ldirectord

然后用下面命令启动集群系统：

/etc/init.d/heartbeat start

4.5 系统测试

4.5.1 测试工具及环境

1 测试工具

本文采用的测试软件是微软的 Web 应用负载测试工具 Web Application Stress Tool(WAS)[56]，它可以以有限的客户端模拟大量的虚拟用户，并发的发送请求到任何采用了 HTTP 标准的服务器，并且不需要考虑服务器运行的平台，然后可以通过相应的参数测试出集群系统的性能。

2 测试环境

本次建构的 LVS 集群测试采用的网络拓扑如图 4-2 所示，其中 RealServer 上被测试的子网为 192.168.1.*，配置 Apache 服务器；Client 模拟 HTTP 用户对 RaelSevrer 进行访问，发送 HTTP 请求，并收集测试结果；LoadBalancer 具有负载均衡的功能，对外的合法 IP 地址为 202.206.151.114，与 RS 相连的内部 IP 地址为 192.168.1.254。LoadBalancer Backup 作为负载均衡器的备份，对外的合法 IP 地址为 202.206.151.114，与 RS 相连的内部 IP 地址为 192.168.1.253，通过心跳线与 LoadBalancer 相连。

3 测试参数

评价集群系统性能好坏的指标有两个：响应时间和吞吐率。前者反映服务质量，响应时间越短，说明服务质量越好；后者反映服务器集群的性能，吞吐率越大，说明单位时间内接受和服务的 HTTP 请求越多。

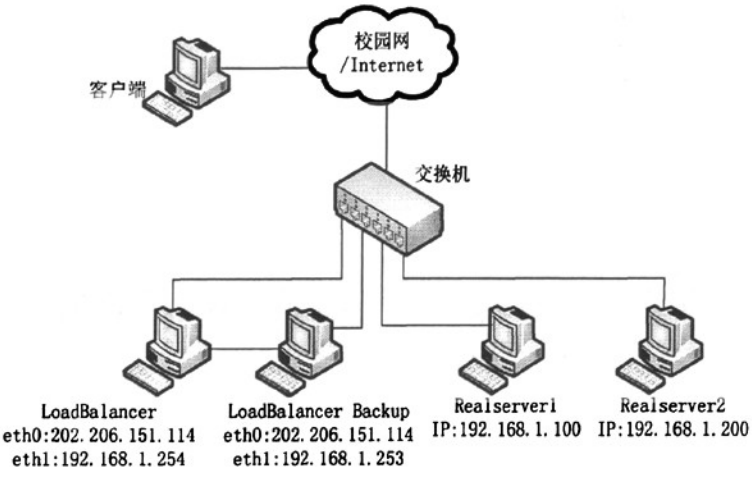

图 4-2 测试集群的网络拓扑图

Figure4-2 Network Topology of Testing Cluster

4.5.2 测试内容

1 算法性能测试

对于评价集群系统性能指标的两个参数：吞吐率和响应时间，在 WAS 中可以分别用 Bytes Revc Rate(简记为 BRR)和 Time to First Byte(简记为 TFB)来表示。BRR 表示客户端单位时间收到的字节数，TFB 是指客户端收到服务器发送的第一个字节的时间。测试时，在客户机上利用 WAS 同时向负载均衡器发送连接数为 100、200、300、400、500、600、700 和 800 的并发请求。图 4-3 是负载均衡器没有采用 LVS 集群，以及在集群中分别采用 WLC 算法和 DF 算法时所测得的 BRR 数据。图 4-4 是负载均衡器没有采用 LVS 集群，以及在集群中分别采用 WLC 算法和 DF 算法时所测得的 TFB 数据。

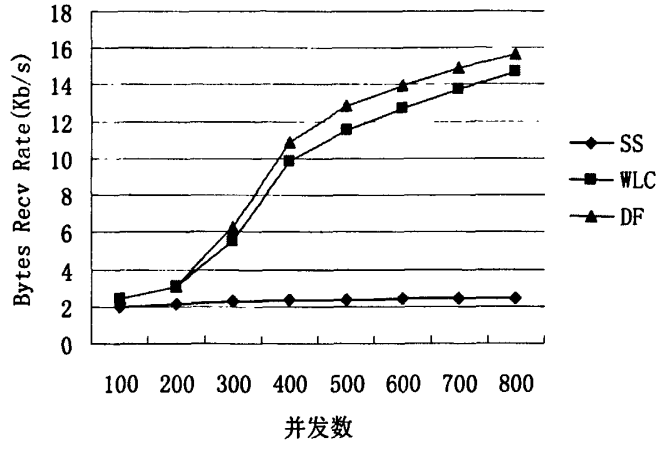

图 4-3 三种情况下 BRR 对比

Figure4-3 BRR Comparison of Single Server、DF and WLC

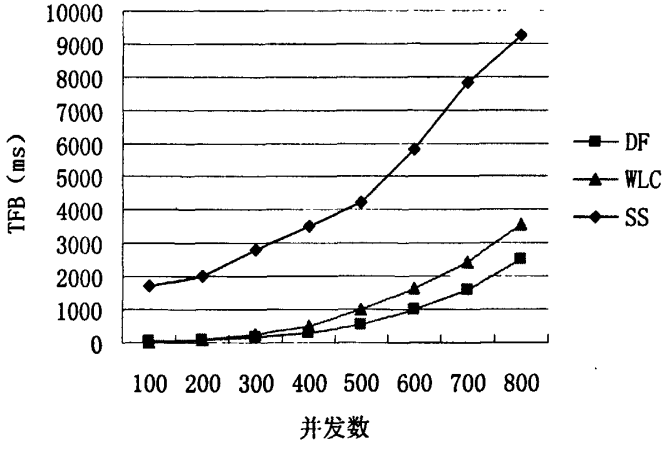

图 4-4 三种情况下 TFB 对比

Figure4-4 TFB Comparison of Single Server、DF and WLC

从图 4-3 和图 4-4 中的数据分析来看，可以得出以下结论：当请求连接数较小时，两种算法效果相似，WLC 算法的响应时间比 DF 的要略小一点，这是因为当连接数较小时，DF 算法定时获取 CPU 利用率和响应时间等负载信息的额外开销影响了其性能。但是随着连接数的增大，DF 算法的优越性就表现出来了，其吞吐率比 WLC 要高，而响应时间相对要小，虽然此时 DF 同样有额外的开销，但是当连接数较大时，这些额外的开销被其产生的效果抵销了。因此，总体来说，DF 算法是有效的，并且在一定连接数的情况下，其效果要优于 WLC 算法。

2 系统容错性能测试

此集群系统具有一定的容错功能。由于系统的动态反馈功能，当服务器池中

的某一台服务器发生故障时，负载均衡调度器无法从其获得负载信息，此时，负载监控进程会将其权值设为 0。此后，调度器将不会将链接发往此服务器，其他服务器将接管所有工作。此外，在某台服务器需要维护时也可利用这个特点，将需要维护的服务器关闭后不会影响系统正常运行。为了测试此功能，我们在系统运行一段时间后，我们将服务器 RS1 关闭，观察关闭后服务器链接分配情况，如图 4-5 所示：

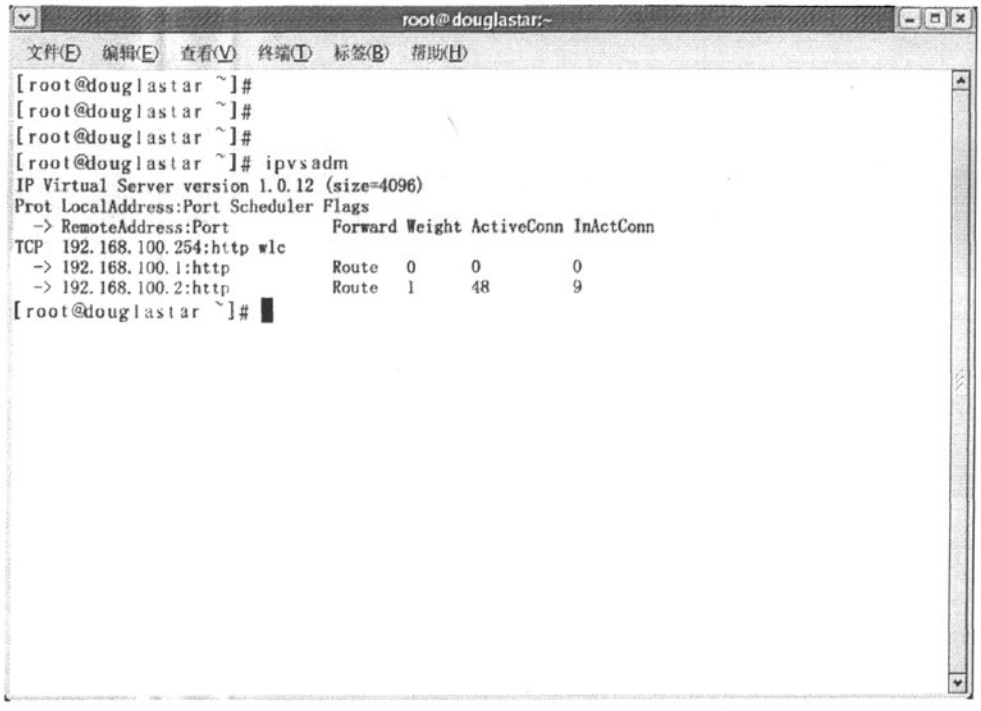

图 4-5 系统容错性能测试

Figure4-5 Test of Performance fault-tolerant

从上图我们可以看出，服务器 RS1 在关闭后，由于负载均衡调度器 LB 不能从其取得负载信息，负载监控进程将其权值设置为 0。因此所有的链接到达调度器时，调度器将它们全部转发到服务器 RS2 上，从而实现了系统的容错保护功能，提高了系统的可用性。

3 系统伸缩性能测试

在系统容量不断增加时，如果原来的服务器台数不能满足需要，可以在不影响原有系统的情况下增加服务器的数量。为测试此功能，在开始我们只使用一台服务器 RSl，然后再加入另一台服务器 RS2，并对负载均衡调度器和 RS2 进行相应设置，观察 RS2 加入前后链接的分配情况，如图 4-6、4-7 所示。

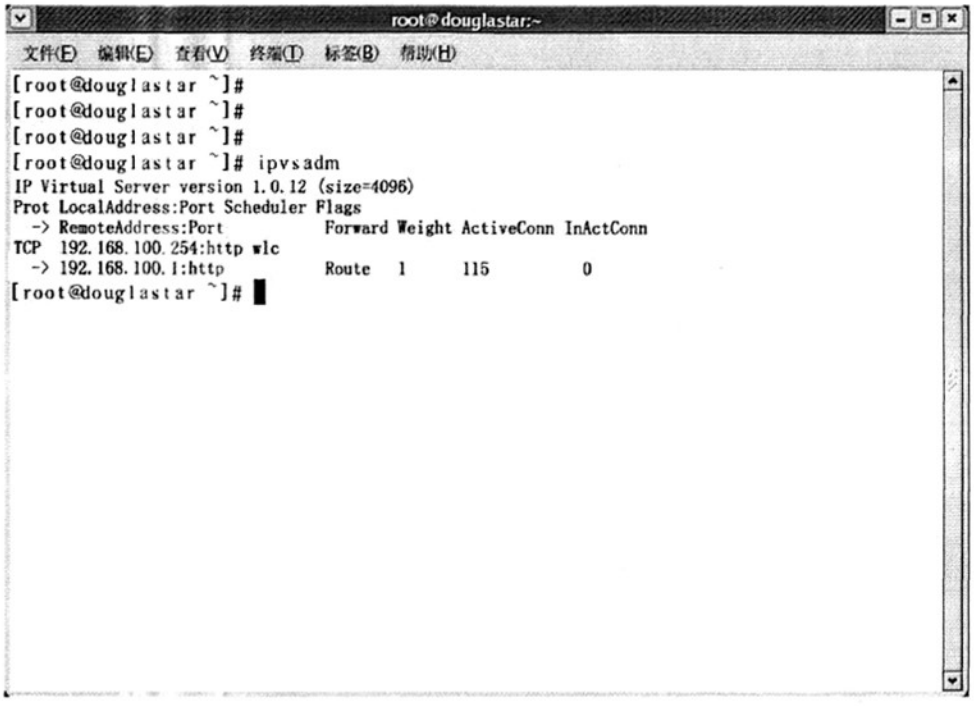

图 4-6 加入服务器 RS2 前的情况

Figure4-6 Situation before adding RS2

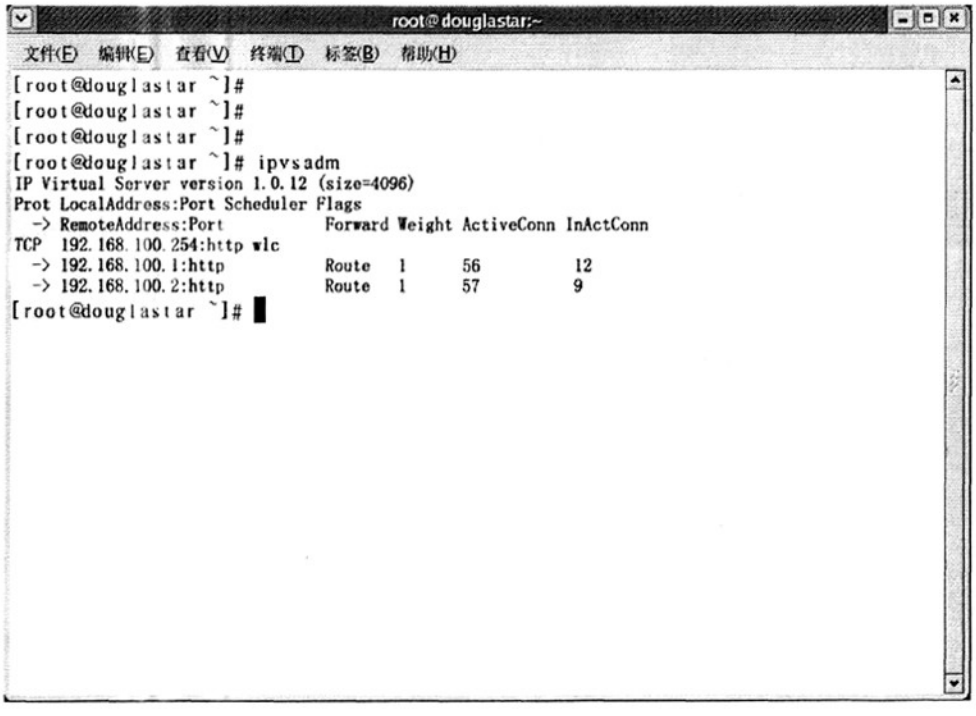

图 4-7 加入服务器 RS2 后的情况

Figure4-7 Situation after adding RS2

由上图可知，在集群系统加入服务器后，调度器会根据负载均衡调度算法将

链接分给新加入的服务器，减轻原有服务器的负载，从而使系统具有很强的伸缩性。

4.6 本章小结

本章分析了现实需求下 Web 服务器集群的设计目标、体系结构，给出了组建集群的具体步骤，构建了一个实际的 Web 服务器集群，并在此平台上实现了改进的负载均衡算法。然后利用微软公司的 WAS 测试工具对该算法的性能进行了测试，对比已有的加权最小连接数算法，分析了测试结果，结果表明改进的算法有较好的性能，同时对集群的容错性和伸缩性进行了测试，结果表明该集群具有较好的容错性和伸缩性，可以满足 Web 服务的需要。

结论

Web 应用是互联网的重要服务之一，随着访问量的不断增加，Web 服务器正变得越来越不堪重负。本文针对我校网络教学服务器存在的因大量请求而引起的负载过重和访问瓶颈问题，对 LVS 集群技术和负载均衡技术进行了认真研究，采用 LVS 集群解决方案，构建了一个 Web 服务器集群，代替原来的单个服务器，提高了服务器系统的性能，发挥了集群系统的高可用性、可伸缩性和良好的性价比等优点。

本文首先分析了 LVS 集群系统的体系结构、工作原理和软件结构及实现细节，研究了基于 IP 层的三种负载均衡技术和八种基本调度算法。在此基础上分析了现有 LVS 负载均衡算法的不足之处，设计并实现了一种动态反馈负载均衡算法。最后，在现有的网络条件下构建了一个 Web 服务器集群，对该服务器集群和动态调度算法的性能进行了测试。测试结果表明，该服务器集群不仅能够解决网络教学服务器的访问瓶颈问题，提高了 Web 服务系统的整体性能，而且提高了服务器的可靠性和可扩展性，起到了较好的负载均衡效果。

服务器集群技术是一个快速发展的领域，这几年越来越受到人们的重视，成为高性能网络服务系统的解决方案之一。受各种因素的限制，本文所设计的算法和构建的集群系统中还存在一些有待改进的地方：

(1)本文构建的服务器集群是基于 VS/NAT 模式的，这种模式在服务器数量不超过 10 台的情况下可以很好的工作，但是当规模再大时，负载均衡器将成为新的瓶颈，因此在条件允许的情况下应该考虑采用效率更高的 VS/DR 模式。

(2)在本文提出的改进算法中，只考虑了后端服务器的 CPU 利用率和请求响应时间这两个影响因素，在今后的研究中，可以进一步将磁盘利用率、内存利用率和网络带宽利用率等因素考虑进来，以便更加准确的反映服务器的负载状况。

(3)受实际条件的限制，本文在对改进的算法进行测试时，所构建的 Web 服务器集群只采用了两台真实服务器。在条件许可的情况下，应在较大规模的 Web 集群系统中对改进算法的性能进行测试，并使用更专业的负载测试工具例如 LoadRunner 进行测试，以便得到更精确的测试数据。

参考文献

1 章文嵩 Linux 服务器集群系统(一),
URL: http://www-128.ibm.com/developerworks/cn/linux/cluster/lvs/part1

2 Linux Virtual Server Project. URL: http://www.linuxvirtualserver.org

3 曾碧卿等.服务器集群系统研究.计算机应用研究,2004.(3): 186-187.

4 Buyya R.高性能集群计算:结构与系统(第一卷).郑纬民、石威、汪东升等译.电子工业出版社,2001:105-106.

5 张小芳,胡正国,郑继川.高可用性集群技术的研究和应用.计算机工程,2003,29(04):26-27.

6 刘键,徐磊,张维明.基于动态反馈的负载均衡算法.计算机工程与科学,Vol.25,No.5, 2003:65-66

7 章文嵩.可伸缩网络设计与实现.国防科学技术大学博士论文.2000:3-7.

8 Wensong Zhang, Shiyao Jin, Quanyuan Wu. Linux Virtual Server: Server Clustering for Scalable Network Service[A].In:World Congress Conference. Proceeding of World congress Conference [C].Beijing, 2000:175-181.

9 VALERIA C, MICHELE C, PHILIPS Y. Dynamic Load Balancing on Web Server Systems. IEEE Internet Computing, 1999, 3(3):pp28-29.

10 Robert Simon, and Aran K.Sood, "End-to-End Analysis of Distributed Video-on-Demand Systems", IEEE TRANSACTIONS ON MULTIMEDIA, VOL.6,NO.1, FEBRUARY :pp282-290.

11 Mohammed A.M.Ibrahim, and Lu Xinda, "Performance of Dynamic Load Balancing Algorithm on Cluster of Workstations and PCs", Proceedings of the Fifth International, Conference on Algorithms and Architecture for Parallel Processing(ICA3PP' 02):pp304-309

12 I. Banicescu, V. Velusamy, J. Devaprasad. On the scalability of dynamic scheduling sciential applications with adaptive weighted factoring. Cluster computing: The Journal of Networks, Software Tools and Applications 6 (3) (2003): 215-226.

13 Iyengar Arun, MacNair Ed, Nguyen Thao. An analysis of Web server performance. In: Proceedings of Global Telecommunications Conference, 1997.3: 1943-1947.

14 Bryhni Haakan. A comparison of load balancing techniques for scalable Web servers. IEEE Network, 2001,(7/ 8):58-64

15 Hwang Suntae, Jung Naksoo. Dynamic scheduling of Web server cluster. In: Proceedings of IEEE 9th International Conference Parallel and Distributed System,2002, 563-568.

16 Li Chuan Chen, Hyeon Ah Choi. Approximation algorithms for data distribution with load balancing of Web servers. In:Proceedings of IEEE International Conference on Cluster

Computing, 2001,274- 281.

17 Casslicchio Emiliano, Tucci Salvatore. Static and Dynamic scheduling algorithm for scalable Web server farm. In:Proceedings of the IEEE 9th Euromicro Workshop on Parallel and Distributed processing, 2001,369-376

18 K. Barker, N. Chrisochoides, An evaluation of a framework for the dynamic Load balancing of highly adaptive and irregular parallel applications. In Proceedings of the IEEE/ACM Supercomputing 2003 Conference, 2003, p.(CDROM)

19 Mohammed.A.M.Ibrahim, Lu Xinda. Utilization of Cluster of PCs for the Study of Dynamic Load Balancing. in proceedings of IEEE TENON' 02, 2002:pp276-280

20 Alxe Vrenios 著.马朝晖等译.Linux 集群体系结构.机械工业出版社，2003:85-87

21 章文嵩 Linux 服务器集群系统(四)，
URL: http://www-128.ibm.com/developerworks/cn/linux/cluster/lvs/part4

22 章文嵩 Linux 服务器集群系统(二)，
URL: http://www-128.ibm.com/developerworks/cn/linux/cluster/lvs/part2

23 YM Teo,R Ayani.Comparsion of Load Balancing Strategies on Cluster-based Web Servers,Transactions of the Society for Modeling and Simulation,2001:108-109

24 张锦祥.基于 LVS 集群服务器的设计和实现.浙江教育学院学报,2005.(5): 60~64

25 杨磊,郭庆平.负载均衡技术分析及 LVS 实现武汉理工大学学报(交通科学与工程版)2004.28(1):77-79

26 程洪,钱乐秋,洪圆.基于 Linux 集群的 Web 服务的研究和构建.计算机工程与应用,2004.(34): 158-161

27 李双庆,古平,程代杰.Web 系统负载均衡策略分析与研究.计算机工程与应用，2002(19):40-43

28 郑灵翔,刘君尧,陈辉煌.Linux 下的负载均衡集群 LVS 实现分析与测试.厦门大学学报(自然科学版).2002.4(16):726-730.

29 马卫.一种改进的 LVS 负载均衡算法. 华中师范大学硕士论文.2004:28-29

30 朱文涛,洪佩琳,李津生.基于 Linux 虚拟服务器的负载均衡.计算机工程. 2002.28(12):55-57

31 V Cardellini P Yu.Dynamic Load Balaneing on Webserver Systems. IEEE Intemet ComPuting 1999

32 TA NGUYEN ZHOU SUIPING. A Dynamic Load Sharing Algorithm for Massively MultiPlayer Oline Games.SydneyAustralia.2003

33 王晋鹏,潘龙法,李降龙.LVS 集群中的动态反馈调度算法.计算机工程.2005,3(19):40-42

34 郑祺 一种基于动态反馈的负载均衡算法研究.计算机时代.2006(2):49-51

35 谢茂涛,宋中山.LVS 集群系统负载均衡策略的研究.计算机工程与科学 2006,28(8):30-31

36 郭全生,舒继武,毛希平,温冬蝉,郑纬民.基于 LVS 系统的负载动态平衡设计与实现.计算机研究与发展.2004,41(6):923-929

37 I. Banicescu, V. Velusamy, J. Devaprasad. On the scalability of dynamic scheduling sciential applications with adaptive weighted factoring. Cluster computing: The Journal of Networks, Software Tools and Applications 6 (3) (2003): 215-226.

38 Trevor Schroeder, Steve Goddard, Byrav Ramamurthy. Scalable Web Server Clustering Technologies.IEEE.Network,2000;56:0890-8044

39 Yi-Hsing Chang, J.Wey Chen. Designing an Enhanced PC Cluster System for Scalable Network Services.Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA' 05)[C]IEEE. 2005: 1550-445X/05

40 Xuehong Gan, Trevor Schroeder, Steve Goddard and Byrav Ramamurthy. LSMAC vs. LSNAT: Scalable cluster-based Web servers. Cluster Computing，2000.3:175-185

41 Jian Liu Longlu Xu Baogen Gu Jing Zhang. A Scalable, High Performance Internet Cluster Server.IEEE.2000: 0-7695-0589-2/00

42 ARUN IYENGAR, JIM CHALLENGER, DANIEL DIAS, PAUL DANTZIG. High Performance Web Site Design Techniques.IEEE.Internet Computing,2000;34:1089-7801

43 Xuehong Gan and Byrav Ramamurthy. LSMAC: An improved load sharing network service dispatcher. World Wide Web 3 (2000) :53–59

44 Liming Liu and Yumao Lu. Dynamic traffic controls for Web-server networks. Computer Networks 45 (2004): 523–536

45 Daniel Andresen, Tao Yang and Oscar H. Ibarra. Toward a Scalable Distributed WWW Server on Workstation Clusters . Journal of Parallel and Distributed Computing，Volume 42, Issue 1, 10 April 1997, Pages 91-100

46 Ahmad Faour and Nashat Mansour. Weblins: A scalable WWW cluster-based server. Advances in Engineering Software, Volume 37, Issue 1, January 2006, Pages 11-19

47 Jiannong Cao, Yudong Sun, Xianbin Wang and Sajal K. Das. Scalable load balancing on distributed web servers using mobile agents . Journal of Parallel and Distributed Computing, Volume 63, Issue 10, October 2003, Pages 996-1005

48 Chun-Wei Tseng and Chu-Sing Yang. System support for web hosting services on server clusters]. Computers & Electrical Engineering, Volume 33, Issue 3, May 2007, Pages 208-220

49 秦刘,兰巨龙,杨帅.动态反馈负载均衡在 LVS 集群中的设计与实现.电子技术应用,2007(9):116-119.

50 陈莉君.深入分析 Linux 内核源代码.人民邮电出版社.2002.08:729-751

51 倪继利.Linux 内核分析及编程.电子工业出版社,2005.09:224-240

52 李玉波,朱自强,郭军.Linux C 编程.清华大学出版社,2005.09:131-143

53 朱建新,朱益峰.利用 LVS 和 NAT 构建高可用的 Linux 负载均衡集群.南通大学学报,2005.04(2): 62~65

54 刘文涛.网络安全开发包详解.电子工业出版社，2005:215-226

55 Zhang W, Jin S, W Q. Scaling Internet services by Linux director. Los Alamitos. IEEE Computer Society. 2000:pp996-1005

56 软件测试工具--WAS 服务器负载测试软件导读.
URL: http://se.csai.cn/casepanel/ST/No024.htm?ID=1267.

攻读硕士学位期间所发表的学术论文

1.杜晓军,耿凯,吴秋红.基于 PXE 协议的 Linux 自动安装原理分析与应用.通信技术,2008,41(08):137-138

致谢

本文的研究工作是在导师张建标老师的悉心指导下完成的。我攻读在职研究生期间，得到了张老师的热情帮助和悉心关怀；论文研究过程中，张老师提出了许多宝贵的意见和建议，使我得以顺利完成学位论文。张老师深厚的理论功底、严谨的治学态度、敏锐的科学洞察力和清晰合理的思维方式给我留下了深刻的印象，必将成为我今后工作和学习的榜样。值此论文完成之际，向张老师表示深深地敬意和感谢！

同时，要感谢计算机学院的全体老师辛勤培养和教诲！在学校脱产学习的一年中，从老师们身上不仅学到了丰富的专业知识，开阔了思想和眼界，更体会到老师们一丝不苟的治学态度、锐意进取的品质和高超的教学艺术，使我受益非浅。

感谢我所在单位的领导和同事，他们给我提供了一个良好的研究和学习环境，对研究中遇到的一些问题给了我很大的帮助。

感谢我的朋友和家人，他们的理解和支持给了我更多的信心，使我能够顺利的完成研究工作。

最后，对在我学习和成长道路上给予帮助的所有老师和朋友们表示深深地感谢，对评阅该论文的所有专家表示最真挚的敬意和感谢！。