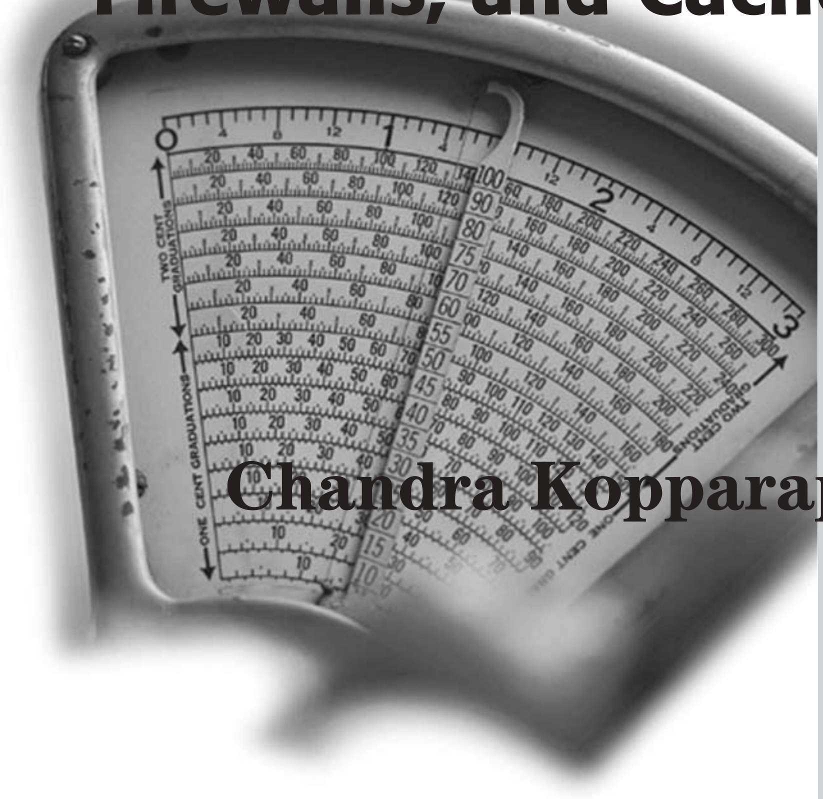
Load Balancing Servers, Firewalls, and Caches



Chandra Kopparapu

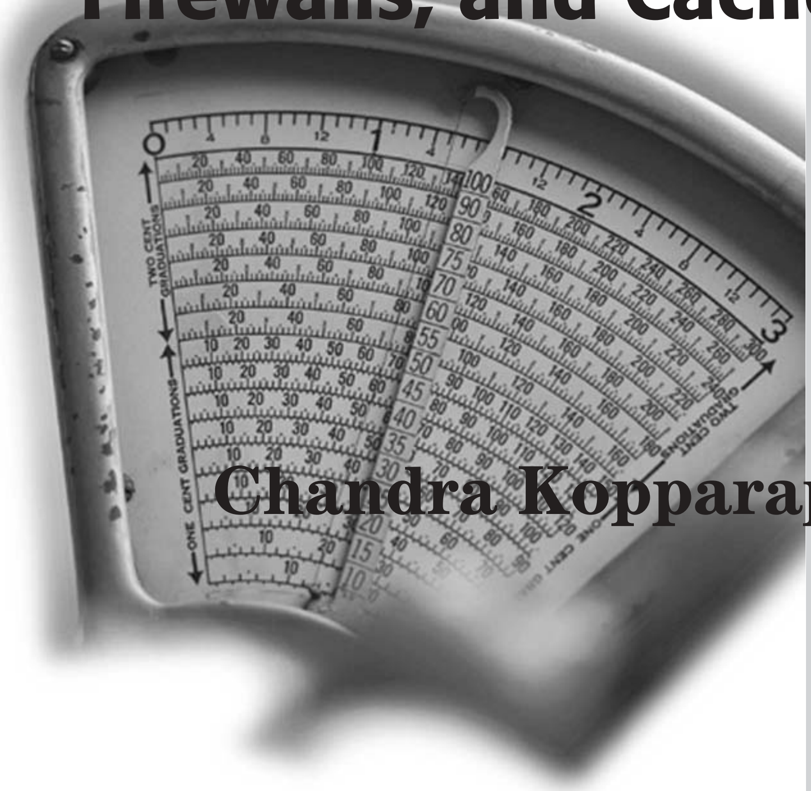
Wiley Computer Publishing

John Wiley & Sons, Inc.

NEW YORK • CHICHESTER • WEINHEIM • BRISBANE • SINGAPORE • TORONTO

Load Balancing Servers, Firewalls, and Caches

Load Balancing Servers, Firewalls, and Caches



Chandra Kopparapu

Wiley Computer Publishing

John Wiley & Sons, Inc.

NEW YORK • CHICHESTER • WEINHEIM • BRISBANE • SINGAPORE • TORONTO

Publisher: Robert Ipsen

Editor: Carol A. Long

Developmental Editor: Adaobi Obi

Managing Editor: Micheline Frederick

Text Design & Composition: Interactive Composition Corporation

This book is printed on acid-free paper. ♾

Published by John Wiley & Sons, Inc.

Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Kopparapu, Chandra.
  Load balancing servers, firewalls, and caches / Chandra Kopparapu.
    p. cm
  Includes bibliographical references and index.
  ISBN 0-471-41550-2 (cloth : alk. paper)
   1. Client/server computing. 2. Firewalls (Computer security) I. Title.

QA76.9.C55 K67 2001
004.6--dc21

2001046757

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To my beloved daughters,
Divya and Nitya,
who bring so much joy to my life.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

First and foremost, my gratitude goes to my family. Without the support and understanding of my wife and encouragement from my parents, this book would not have been completed.

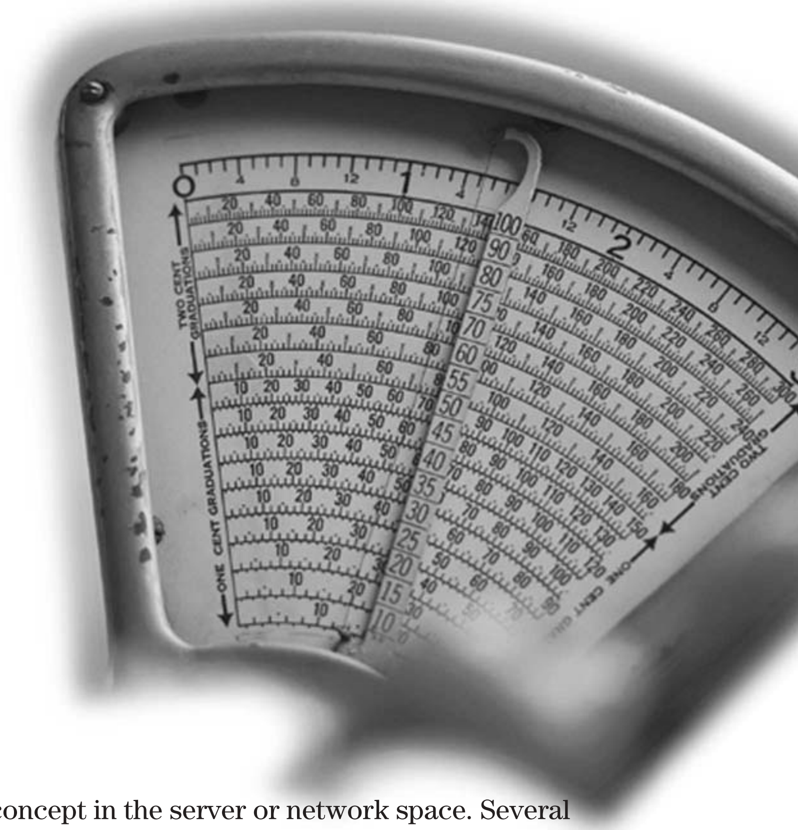
Rajkumar Jalan, principal architect for load balancers at Foundry Networks, was of invaluable help to me in understanding many load-balancing concepts when I was new to this technology. Many thanks go to Matthew Naugle, systems engineer at Foundry Networks, for encouraging me to write this book, giving me valuable feedback, and reviewing some of the chapters. Matt patiently spent countless hours with me, discussing several high-availability designs, and contributed valuable insight based on several customers he worked with. Terry Rolon, who used to work as a systems engineer at Foundry Networks, was also particularly helpful to me in coming up to speed on load-balancing products and network designs.

I would like to thank Mark Hoover of Acuitive Consulting for his thorough review and valuable analysis on Chapters 1, 2, 3, and 9. Mark has been very closely involved with the evolution of load-balancing products as an industry consultant and guided some load-balancing vendors in their early days. Many thanks to Brian Jacoby from America Online, who reviewed many of the chapters in this book from a customer perspective and provided valuable feedback.

Countless thanks to my colleagues at Foundry Networks, who worked with me over the last few years in advancing load-balancing product functionality and designing customer networks. I worked with many developers, systems engineers, customers, and technical support engineers to gain valuable insight into how load balancers are deployed and used by customers. Special thanks to Srini Ramadurai, David Cheung, Joe Tomasello, Ivy Hsu, Ron Szeto, and Ritesh Rekhi for helping me understand various aspects of load balancing functionality. I would also like to thank Ken Cheng, VP of Marketing at Foundry, for being supportive of this effort, and Bobby Johnson, Foundry's CEO, for giving me the opportunity to work with Foundry's load-balancing product line.

CHAPTER 1

Introduction



Load balancing is not a new concept in the server or network space. Several products perform different types of load balancing. For example, routers can distribute traffic across multiple paths to the same destination, balancing the load across different network resources. A server load balancer, on the other hand, distributes traffic among server resources rather than network resources. While load balancers started with simple load balancing, they soon evolved to perform a variety of functions: load balancing, traffic engineering, and intelligent traffic switching. Load balancers can perform sophisticated health checks on servers, applications, and content to improve availability and manageability. Because load balancers are deployed as the front end of a server farm, they also protect the servers from malicious users, and enhance security. Based on information in the IP packets or content in application requests, load balancers make intelligent decisions to direct the traffic appropriately—to the right data center, server, firewall, cache, or application.

The Need for Load Balancing

There are two dimensions that drive the need for load balancing: servers and networks. With the advent of the Internet and intranet, networks connecting the servers to computers of employees, customers, or suppliers have become mission critical. It's unacceptable for a network to go down or exhibit poor performance, as it virtually shuts down a business in the Internet economy. To build a Web site for e-commerce, for example, there are several components that must be looked at: edge routers, switches, firewalls, caches, Web servers, and database servers. The proliferation of servers for various applications has created data centers full of server farms. The complexity and challenges in scalability, manageability, and availability of server farms is one driving factor behind the need for intelligent switching. One must ensure scalability and high availability for all components, starting from the edge routers that connect to the Internet, all the way to the database servers in the back end. Load balancers have emerged as a powerful new weapon to solve many of these issues.

The Server Environment

There is a proliferation of servers in today's enterprises and *Internet Service Providers* (*ISPs*) for at least two reasons. First, there are many applications or services that are needed in this Internet age, such as Web, FTP, DNS, NFS, e-mail, ERP, databases, and so on. Second, many applications require multiple servers per application because one server does not provide enough power or capacity. Talk to any operations person in a data center, and he or she will tell you how much time is spent in solving problems in manageability, scalability, and availability of the various applications on servers. For example, if the e-mail application is unable to handle the growing number of users, an additional e-mail server must be deployed. The administrator must also think about how to partition the load between the two servers. If a server fails, the administrator must now run the application on another server while the failed one is repaired. Once it has been repaired, it must be moved back into service. All of these tasks affect the availability and/or performance of the application to the users.

The Scalability Challenge

The problem of scaling computing capacity is not a new one. In the old days, one server was devoted to run an application. If that server did not do the job, a more powerful server was bought instead. The power of servers grew as different components in the system became more powerful. For example, we saw the processor speeds double roughly every 18 months—a phenomenon now known as Moore's law, named after Gordon Moore of Intel Corporation. But the demand for computing grew even faster. Clustering technology was therefore invented, originally for mainframe computers. Since mainframe computers were proprietary, it was easy for mainframe vendors to use their own technology to deploy a cluster of mainframes that shared the computing task. Two main approaches are typically found in clustering: loosely coupled systems and symmetric multiprocessing. But both approaches ran into limits, and the price/performance is not as attractive as one traverse up the system performance axis.

Loosely Coupled Systems

Loosely coupled systems consist of several identical computing blocks that are loosely coupled through a system bus or interconnection. Each computing block contains a processor, memory, disk controllers, disk drives, and network interfaces. Each computing block, in essence, is a computer in itself. By gluing together a multiple of those computing blocks, vendors such as Tandem built systems that housed up to 16 processors in a single system. Loosely coupled systems use interprocessor communication to share the load of a computing task across multiple processors.

Loosely coupled processor systems only scale if the computing task can be easily partitioned. For example, let's define the task as retrieving all records from a table that has a field called *Category Equal to 100*. The table is partitioned into four equal parts, and each part is stored in a disk partition that is controlled by one processor. The query is split into four tasks, and each processor runs the query in parallel. The results are then aggregated to complete the query.

However, not every computing task is that easy. If the task were to update the field that indicates how much inventory of lightbulbs are left, only the processor that owns the table partition containing the record for lightbulbs can perform the update. If sales of lightbulbs suddenly surged, causing a momentary rush of requests to update the inventory, the processor that owned the lightbulbs record would become a performance bottleneck, while the other processors would remain idle. In order to get the desired scalability, loosely coupled systems require a lot of sophisticated system and application level tuning, and need very advanced software, even for those tasks that can be partitioned. Loosely coupled systems cannot scale for tasks that are not divisible, or for random hot spots such as lightbulb sales.

Symmetric Multiprocessing Systems

Symmetric multiprocessing (SMP) systems use multiple processors sharing the same memory. The application software must be written to run in a multithreaded environment, where each thread may perform one atomic computing function. The threads share the memory and rely on special communication methods such as semaphores or messaging. The operating system schedules the threads to run on multiple processors so that each can run concurrently to provide higher scalability. The issue of whether a computing task can be cleanly partitioned to run concurrently applies here as well. As processors are added to the system, the operating system needs to work more to coordinate among different threads and processors, and thus limits the scalability of the system.

The Network Environment

Traditional switches and routers operate on IP address or MAC address to determine the packet destinations. However, they can't handle the needs of complex modern server farms. For example, traditional routers or switches cannot intelligently send traffic for a particular application to a particular server or cache. If a destination server is down, traditional switches continue sending the traffic into a dead bucket. To understand the function of traditional switches and routers and how Web switching represents advancement in the switching technology, we must examine the *Open Systems Interface (OSI)* model first.

The OSI Model

The OSI model is an open standard that specifies how different devices or computers can communicate with each other. As shown in Figure 1.1, it consists of seven layers, from physical layer to application layer. Network protocols such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Protocol (IP), and Hypertext Transfer Protocol (HTTP) can be mapped to the OSI model in order to understand the purpose

Layer 7	Application Layer	HTTP, FTP, SNMP, Telnet, DNS
Layer 6	Presentation Layer	
Layer 5	Session Layer	
Layer 4	Transport Layer	TCP, UDP
Layer 3	Network Layer	IP
Layer 2	Data Link Layer	
Layer 1	Physical Layer	

Figure 1.1 The OSI specification for network protocols.

and functionality of each protocol. IP is a Layer 3 protocol, whereas TCP and UDP function at Layer 4. Each layer can talk to its peer on a different computer, and exchange information to the layer immediately below or above itself.

Layer 2/3 Switching

Traditional switches and routers operate at Layer 2 and/or Layer 3; that is, they determine how a packet must be processed and where a packet should be sent based on the information in the Layer 2/3 header. While Layer 2/3 switches do a terrific job at what they are designed to do, there is a lot of valuable information in the packets that is beyond the Layer 2/3 headers. The question is, How can we benefit by having switches that can look at the information in the higher-layer protocol headers?

Layer 4 through 7 Switching

Layer 4 through 7 switching basically means switching packets based on Layer 4–7 protocol header information contained in the packets. TCP and UDP are the most important Layer 4 protocols that are relevant to this book. TCP and UDP headers contain a lot of good information to make intelligent switching decisions. For example, the HTTP protocol used to serve Web pages runs on TCP port 80. If a switch can look at the TCP port number, it may be able to prioritize it or block it, or redirect or forward it to a particular server. Just by looking at TCP and UDP port numbers, switches can recognize traffic for many common applications, including HTTP, FTP, DNS, SSL, and streaming media protocols. Using TCP and UDP information, Layer 4 switches can balance the request load by distributing TCP or UDP connections across multiple servers.

The term *Layer 4–7 switch* is part reality and part marketing hype. Most Layer 4–7 switches work at least at Layer 4, and many do provide the ability to look beyond Layer 4—exactly how many and which layers above Layer 4 a switch covers will vary product to product.

Load Balancing: Definition and Applications

With the advent of the Internet, the network now occupies center stage. As the Internet connects the world and the intranet becomes the operational backbone for businesses, the IT infrastructure can be thought of as two types of equipment: computers that function as a client and/or a server, and switches/routers that connect the computers. Conceptually, load balancers

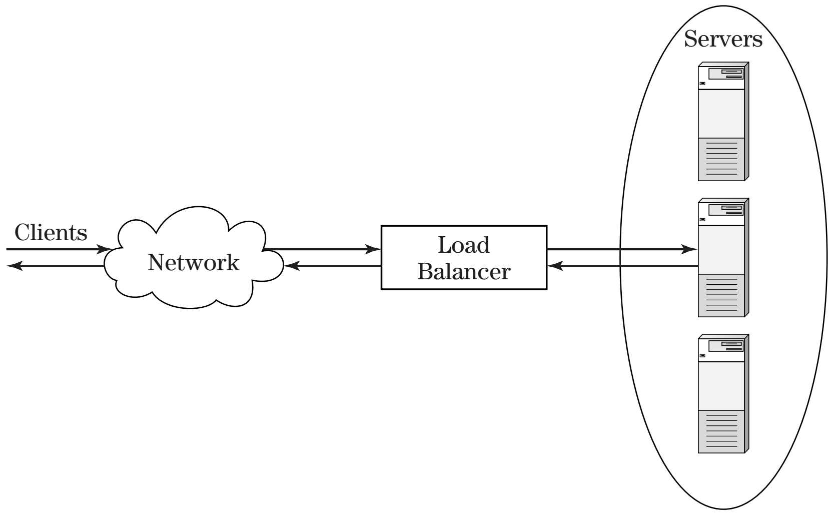


Figure 1.2 Server farm with a load balancer.

are the bridge between the servers and the network, as shown in Figure 1.2. On one hand, load balancers understand many higher-layer protocols, so they can communicate with servers intelligently. On the other, load balancers understand networking protocols, so they can integrate with networks effectively.

Load balancers have at least four major applications:

- Server load balancing
- Global server load balancing
- Firewall load balancing
- Transparent cache switching

Server load balancing deals with distributing the load across multiple servers to scale beyond the capacity of one server, and to tolerate a server failure. Global server load balancing deals with directing users to different data center sites consisting of server farms, in order to provide users with fast response time and to tolerate a complete data center failure. Firewall load balancing distributes the load across multiple firewalls to scale beyond the capacity of one firewall, and tolerate a firewall failure. Transparent cache switching transparently directs traffic to caches to accelerate the response time for clients or improve the performance of Web servers by offloading the static content to caches.

Load-Balancing Products

Load-balancing products are available in many different forms. They can be broadly divided into three categories: software products, appliances, and switches. Descriptions of the three categories follow:

- Software load-balancing products run on the load-balanced servers themselves. These products execute algorithms to coordinate the load-distribution process among them. Examples of such products include products from Resonate, Rainfinity, and Stonebeat.
- Appliances are black-box products that include the necessary hardware and software to perform Web switching. The box may be as simple as a PC or a server, packaged with some special operating system and software or a proprietary box with custom hardware and software. F5 Networks and Radware, for example, provide such appliances.
- Switches extend the functionality of a traditional Layer 2/3 switch into higher layers by using some hardware and software. While many vendors have been able to fit much of the Layer 2/3 switching into ASICs, no product seems to build all of Layer 4–7 switching into ASICs, despite all the marketing claims from various vendors. Most of the time, such products only get some hardware assistance, while a significant portion of the work is still done by software. Examples of switch products include products from Cisco Systems, Foundry Networks, and Nortel Networks.

Is load balancing a server function or a switch function? The answer to this question is not that important or interesting. A more important question is, which load-balancer product or product type better meets your needs in terms of price/performance, feature set, reliability, scalability, manageability, and security? This book will not endorse any particular product or product type, but will cover load-balancing functionality and concepts that apply whether the load-balancing product is software, an appliance, or a switch.

The Name Conundrum

Load balancers have many names: Layer 2 through 7 switches, Layer 4 through 7 switches, Web switches, content switches, Internet traffic management switches or appliances, and others. They all perform essentially similar jobs, with some degree of variation in functionality. Although *load balancer* is a descriptive word, what started as load balancing evolved to encompass much more functionality, causing some to use the term *Web switches*. This book uses the term *load balancers*, because it's a very short and quite descriptive phrase. No matter which load-balancer application we look at, load balancing is the foundation.

How This Book Is Organized

This book is organized into nine chapters. While certain basic knowledge of networking and Internet protocols is assumed, a quick review of any concept critical to understanding the functionality of load balancers is usually provided.

Chapter 1 introduces the concepts of load balancers and explains the rationale for the advent of load balancing. It includes the different form factors of load-balancing products and major applications for load balancing.

Chapter 2 explains the basics of server load balancing, including a packet flow through a load balancer. It then introduces the different load-distribution algorithms, server-and-application health checks, and the concept of direct server return. Chapter 2 also introduces Network Address Translation (NAT), which forms the foundation in load balancing. It is highly recommended that readers unfamiliar with load-balancing technology read Chapters 2, 3, and 4 in consecutive order.

Chapter 3 introduces more advanced concepts in server load balancing, such as the need for session persistence and different types of session-persistence methods. It then introduces the concept of Layer 7 switching or content switching, in which the load balancer directs the traffic based on the URLs or cookies in the traffic flows.

Chapter 4 provides extensive design examples of how load balancers can be used in the networks. This chapter not only shows the different designs possible, but it also shows the evolution of the design and why a particular design is a certain way. This chapter addresses the need for high availability, including designs that tolerate the failure of a load balancer.

Chapter 5 introduces the concept of global server load balancing and the various methods for global server load balancing. This chapter includes a quick refresher of Domain Name Server (DNS) and how it is used in global server load balancing.

Chapter 6 describes how load balancers can be used to improve the scalability, availability, and manageability of firewalls. It also addresses various high-availability designs for firewall load balancing.