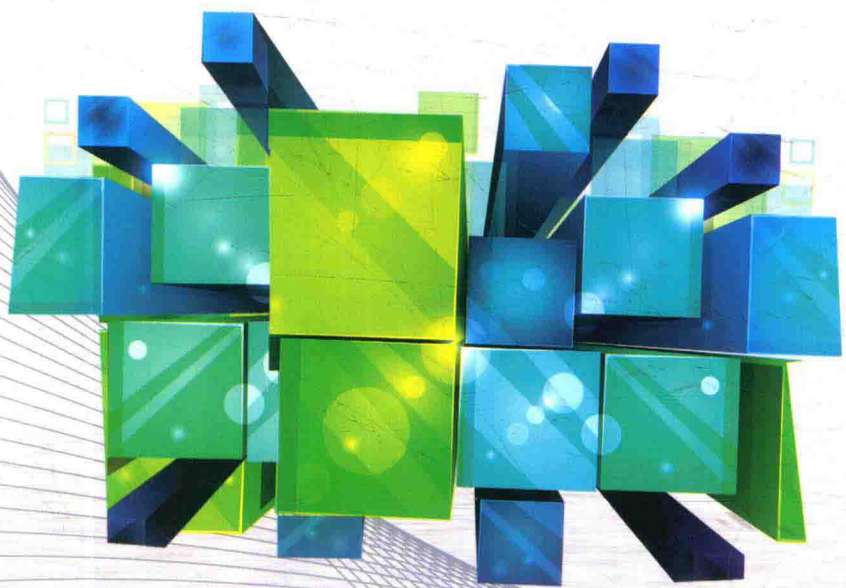
高等学校电子信息类“十三五”规划教材
CDIO工程教育计算机专业实战系列教材

软件测试技术实战

主　编　王铁军
副主编　陈海宁
参　编　邹茂扬　李晓莉



西安电子科技大学出版社
http://www.xduph.com

高等学校电子信息类“十三五”规划教材

CDIO 工程教育计算机专业实战系列教材

软件测试技术实战

主　编　王铁军

副主编　陈海宁

参　编　邹茂扬　李晓莉

西安电子科技大学出版社

内 容 简 介

本书系统地介绍了软件测试的基本概念、测试技术及测试工具，并通过测试实例详细说明了软件测试流程和测试工具的使用。本书采用案例教学法编写，书中提供了软件测试实践案例及相关源代码，以帮助读者增强对软件测试相关知识的融会贯通，快速掌握软件测试技术。

本书结构规范、实例丰富，理论与实践相结合，深入浅出、通俗易懂。全书从软件测试概述和测试环境搭建入手，从代码覆盖测试、单元测试、黑盒测试、负载测试和移动终端测试五个方面讲解了软件测试的实践案例，以供读者全面了解软件测试的整个过程。

本书适合作为普通高等院校计算机相关专业软件测试课程的教材，也可作为软件测试培训班的教材，同时还可作为有志于从事软件测试工作的学生和刚就业人员的入门参考书。

图书在版编目(CIP)数据

软件测试技术实战 / 王铁军主编. —西安：西安电子科技大学出版社，2018.8

ISBN 978-7-5606-4959-7

Ⅰ. ① 软… Ⅱ. ① 王… Ⅲ. ① 软件—测试 Ⅳ. ① TP311.55

中国版本图书馆 CIP 数据核字(2018)第 145181 号

策划编辑 李惠萍

责任编辑 马 凡 阎 彬

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话 (029)88242885 88201467 邮 编 710071

网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com

经 销 新华书店

印刷单位 陕西利达印务有限责任公司

版 次 2018 年 8 月第 1 版 2018 年 8 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印 张 11.5

字 数 244 千字

印 数 1～3000 册

定 价 27.00 元

ISBN 978 - 7 - 5606 - 4959 - 7/TP

XDUP 5261001-1

如有印装问题可调换

中国电子教育学会高教分会推荐

高等学校电子信息类“十三五”规划教材

CDIO 工程教育计算机专业实战系列教材

编审专家委员会名单

主　　任　何　嘉

副 主 任　魏　维

编审人员（排名不分先后）

方　睿　吴　锡　王铁军　邹茂扬　李莉丽

廖德钦　鄢田云　黄　敏　杨　昊　陈海宁

张　欢　徐　虹　李　庆　余贞侠　叶　斌

卿　静　文　武　李晓莉

前言

软件测试是保障软件系统功能、性能、安全性等软件质量因素的重要手段。随着计算机系统和互联网的蓬勃发展，软件系统自身的复杂度在不断提升，同时用户和企业对缩短软件开发周期的需求也变得越来越迫切，这些因素都将导致软件测试工作变得更加困难。

本书以作者长期从事本科生软件测试课程教学的讲义为基础，由多位高校一线教师和软件测试公司项目负责人合作编写，旨在通过案例式教学方式，从软件测试概述和测试环境搭建入手，从代码覆盖测试、单元测试、黑盒测试、负载测试和移动终端测试五个方面向读者介绍常用的软件测试技术，以帮助读者快速掌握相关测试工具的使用方法和技巧。在阅读本书部分章节前，要求读者具备一定的 Java 语言编程基础知识。

本书具有如下特点：

- **以实践为基础**。本书在介绍理论知识的基础上，提供了丰富的实战案例、源代码和参考结果，方便读者在实践中学习。
- **覆盖面广**。本书覆盖了常用的各种软件测试技术和工具，对移动终端这个很少涉及的测试领域进行了举例说明，给出了移动 App 的测试方法。
- **启发式教学**。本书在每一章的最后都给出了一定量的思考题。从这些题目入手，读者可以对书中的知识点进行深入思考和总结提高。
- **通俗易懂**。本书在编写过程中，充分考虑到初级层次的读者水平，尽量以浅显易懂的语言描述了相对深奥的软件测试知识，语言通俗易懂，适合各层次学生和专业人士选用。

本书由王铁军担任主编，陈海宁担任副主编，参加本书编写的还有邹茂扬、李晓莉等。特别感谢西安电子科技大学出版社李惠萍对本书编写所提出的宝贵意见，使得本书得以不断改进和完善。

按照编写目标，编者进行了许多思考和努力，但由于编者水平有限，书中可能还有疏漏和不妥之处，恳请读者批评指正，以便我们不断改进。编者联系邮箱：tjw@cuit.edu.cn。

编　者

2018 年 4 月

目　录

第 1 章　软件测试概述

1.1　软件测试过程

通常，我们可以将被测目标系统看做一个函数：

$$Y=f(X)$$

给定一个已知输入 X，就有一个预期输出 Y，即在测试执行前，根据需求就已经预测到输出结果。每一项需求至少需要两个测试用例：一个正检验，一个负检验。正检验是给定正确的输入 X，测试用例会调用被测试目标系统，得到一个实际的输出 Y'，最后通过判断确定程序得到的 Y' 与预期输出的 Y 是否相同。负检验是给定错误的输入 X'，检验被测目标系统是否会输出相应的结果。图 1-1 描述了这样的测试过程。

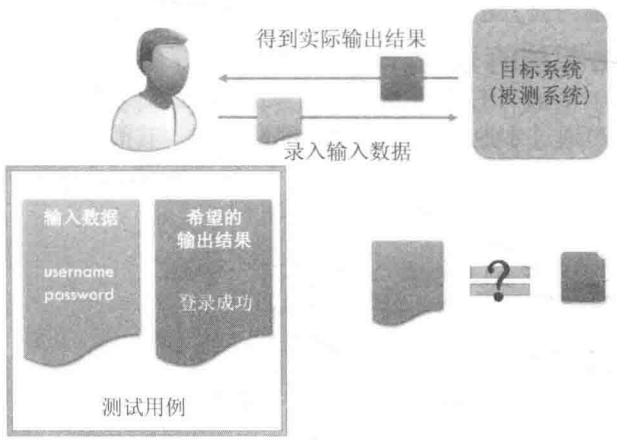


图 1-1　测试过程

1.2　被测目标系统

常见的被测目标系统可以是 Web 系统、应用程序、单机应用、多机应用、游戏及移动应用。针对不同的被测目标系统，可能需要使用不同的测试方法和测试工具对其进行有针

对性的测试。

本书将涉及单机应用、Web 系统和移动 App 这三类被测目标系统。其中应用最为广泛的是 Web 系统的测试，因此本书也会安排主要篇幅来讲解 Web 系统的测试。

1.2.1 Web 系统简介

Web 系统也叫 Web 应用程序(Web Application)，是一种可以通过 Web 方式访问的应用程序。如图 1-2 所示，相比较基于 C/S 架构的应用程序，基于 B/S 架构的 Web 系统的最大优势是，用户不需要在本地安装客户端，只需要有浏览器即可通过互联网访问被测目标系统(即网站系统)。

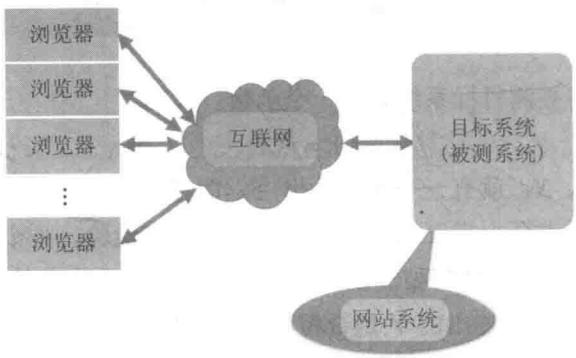


图 1-2 Web 系统的访问模式

1.2.2 用户与 Web 系统的交互

通常情况下，用户通过本地的浏览器访问网站系统，其间主要使用的是 HTTP 协议，如图 1-3 所示。

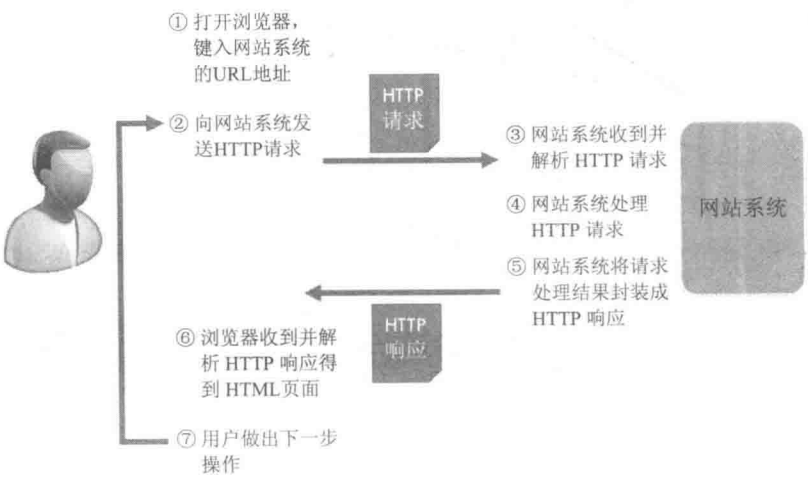


图 1-3 用户使用 HTTP 协议访问网站系统

首先，用户需要打开浏览器，并在浏览器地址栏中输入远端网站系统的 URL 地址。浏览器解析用户输入的 URL 地址，并向网站系统发送 HTTP 请求。网站系统收到用户发送的 HTTP 请求之后，会对收到的 HTTP 请求进行解析，解析后会将其传递给适当的进程进行处理(如从数据库中查找符合条件的数据)。多数情况下，网站系统负责处理用户请求的进程会将处理完的结果以 HTML 形式进行封装(如在浏览器中看到的表格)，并将其封装在 HTTP 响应报文中，通过互联网发送给浏览器。浏览器在收到 HTTP 响应之后，对获取到的 HTML 页面进行解析，并显示给用户。最后，用户在浏览器中即可看到请求得到的结果，并可以进行下一次请求。

上面是用户通过网络与 Web 系统进行交互的典型过程。大多数情况下，Web 服务将同时接收到来自多个用户的多个请求，因此，Web 服务器需要通过一些技术来区分收到的多个请求中，哪些请求来自同一个用户。这些在不同时间按顺序到达 Web 服务器的、来自同一个用户的多个请求，就构成了一个会话(Session)。当有多个用户同时访问 Web 服务时，Web 服务就需要维护与多个用户的会话，进而保证用户与 Web 服务间通信的连贯性与安全性。

1.2.3　Web 系统的演进

在 Web 系统出现之前，多数系统都是基于 C/S(客户端/服务器)架构的模式运行的。并且，由于业务规模不大，基本以单机形式加以部署，即仅由一台服务器提供所有服务。正是由于这样的历史原因，导致最初的 Web 系统也是以单机形式提供服务的。随着系统用户数量的增加、业务自身更加复杂，以及伴随着接入网络速度的提升和用户体验需求的增强，Web 服务模式也经历了从单机到多机，再到集群方式的变迁，如图 1-4 所示。同时，随着 Web 系统服务器数量的增加，高性能、高可用、高伸缩等特性也显得越来越急迫和必需。

图 1-4　Web 服务发展的过程

因此，一个成熟的大型 Web 系统(如 Facebook、淘宝、腾讯、百度等)的系统架构的发展，也是伴随着系统用户量的增加、业务功能的扩展逐渐演变完善的，并且在这个过程中，系统的开发模式、技术架构及设计思想也发生了很大的变化，就连技术人员也从最初的几个人发展到一个部门甚至一条产品线。所以成熟的系统架构是随着业务的扩展而逐步完善的，并不是一蹴而就。不同业务特征的系统，会有各自的侧重点，例如，Facebook 最初侧重的是海量图片的存储和访问，后来逐步发展出消息传递、人脸识别、视频播放等业务需求；虽然淘宝也要处理海量商品的照片信息，但淘宝更需要处理海量商品信息的搜索、下单、支付等业务流程，这些是 Facebook 所不需要的；而腾讯主要解决数亿用户实时消息传

输的问题；百度则要处理海量的搜索请求。由这些实例可以发现，不同的成熟 Web 系统都会因各自业务特性的不同，而进化成不同的系统架构。下面将介绍一个典型的大型 Web 系统的演化过程来帮助读者认识 Web 系统的演进。

1. 单机 Web 系统

正如 1.2.2 节所介绍的，一个 Web 系统要和用户进行交互，首先要能够解析用户发送过来的请求(通常请求以 HTTP 方式封装，当然也会有其他协议，如腾讯 QQ 用的是即时通信协议)，其次要能够对请求进行响应(即处理该请求的业务处理程序)，我们将实现此类功能的组件称为应用程序(Application)，更准确地讲是 Web 应用程序。

最开始的 Web 应用程序需要开发者实现协议解析、会话管理、业务处理等全部功能。后来，随着 Web 应用需求的增加，出现了由特定公司或组织开发和维护的仅仅负责协议解析和会话管理功能的 Web 容器，如微软的 IIS、BEA 的 Weblogic、IBM 的 WebSphere、Apache 的 Tomcat 等。Web 容器的出现，促进了 Web 开发框架的发展，如 Structs、Spring 等，极大地方便了开发者。这样一来，开发者仅需要在特定的 Web 容器上选用某种开发框架，编写满足特定业务逻辑的程序代码，即可完成 Web 应用程序的开发。

在处理业务请求时，另一个必不可少的组件是数据库，如 Oracle、SQL Server、MySQL 等，它们可以提供可靠的数据存储功能和快速的数据查询功能。业务代码仅需要使用标准的 SQL 语句即可访问数据库，完成业务数据的存储和查询。我们把业务数据存放在数据库的过程称为数据持久化。由于最初的数据库都是关系型数据库，与 Java、C++、Python 等面向对象的语言使用方法不同，所以后来出现了专门的数据持久化层，用户可以使用同样的面向对象的方法实现对数据的存储、查询和更新等操作。

此外，还有一部分业务数据，如声音、视频、图片、文档等不适合存放在数据库中，所以需要将这些信息以文件的形式存放到磁盘上。

上述 Web 应用程序、数据库和文件三个组件由于最初业务规模较小，所以，只需要将其部署在同一台服务器上即可满足需求，如图 1-5 所示。

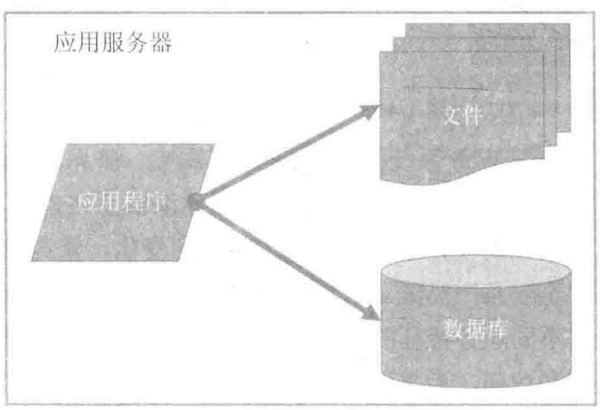


图 1-5　单机架构的 Web 系统

2. 多机 Web 系统

随着业务的扩展及系统用户数量的增加，企业对 Web 系统业务处理能力的需求也随之提高。原有的一台服务器已经不能满足企业对性能的需求，故将应用程序、数据库、文件部署在各自独立的服务器上，如图 1-6 所示。

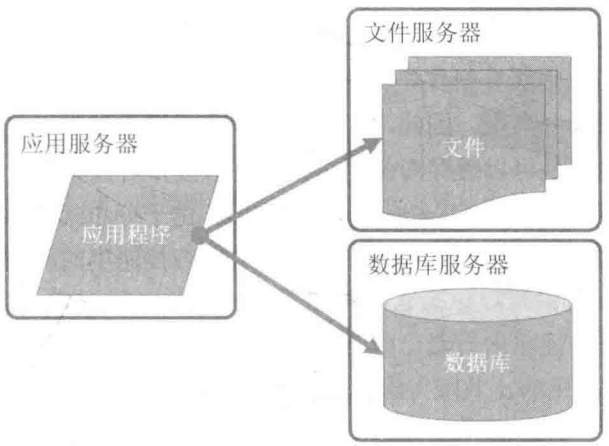


图 1-6　多机架构的 Web 系统

Web 系统的实施人员可根据服务器的用途选择配置不同的硬件，以达到最佳的性能效果。例如：部署 Web 应用程序的应用服务器，可能需要更高处理能力的 CPU 和更大的内存；数据库服务器需要更大的内存和磁盘空间；文件服务器需要更快的网络和更大的磁盘空间。这种分立的服务器架构，为企业定制业务需求提供了保障。并且，随着业务的发展，发现需要存放更多的文件信息时，企业也仅需要升级文件服务器即可，这样在提高 Web 系统灵活性的同时，也为企业节约了维护成本。

3. 应用缓存技术

最初的 Web 系统仅提供信息的发布，很少有用户和 Web 系统的信息交互，因此仅用静态 Web 系统即可实现所需功能，但是随着技术的发展，用户对 Web 系统的需求在不断提高。用户希望 Web 系统可以提供更为及时的信息更新功能和更为方便的信息存储功能，所以动态 Web 系统应运而生，这之后 Web 系统会根据不同的用户基础信息、不同的请求，在数据库中进行查询，根据查询结果动态地生成特定的 Web 页面，并返回给用户。典型的应用包括邮件服务、任务列表、购物车、新闻等。

同样地，随着用户数量的增加，为不同用户动态生成这种独一无二的 Web 页面，并且在每次用户登录或发起请求时都需要重新生成的方式，会给后台的 Web 服务器和数据库服务器带来沉重的压力。统计数据也显示，大多数重新生成的 Web 页面与之前存在的 Web 页面相同，或者多个用户需要看到的 Web 页面是相同的，因此，这些页面就可在生成后被存储到某一个特殊位置，在需要的时候直接读取，这样一来不但减少了后台服务器的压力，也缩短了用户等待的时间，提高了用户体验度。这一技术即为**缓存技术**。

在硬件性能优化的同时，也可通过软件进行性能优化，在大部分网站系统中，都会利用缓存技术改善系统的性能，使用缓存技术主要由于热点数据的存在，大部分网站访问都遵循 28 原则(即 80%的访问请求，最终落在 20%的热点数据上)，所以我们可以对热点数据进行缓存，简化这些数据的访问路径，提高用户体验度。

实现缓存常见的方式有本地缓存和分布式缓存。本地缓存顾名思义是将需要被缓存的数据存放在应用服务器本地，可以存放在内存中，也可以存放在应用服务器的文件系统中，OSCache 就是常用的本地缓存组件。本地缓存的优点是速度快，但由于本地空间有限，所以能够缓存的数据量十分有限。因此，可以使用分布式缓存技术以提高可以使用的缓存空间。分布式缓存的优点是可以缓存海量的数据，并且非常容易扩展，在门户类网站中经常被使用，常用的分布式缓存有 Memcached 和 Redis。由于分布式缓存需要将数据缓存在多个服务器上，所以需要解决缓存数据的一致性和加快查找速度的问题。分布式缓存技术如图 1-7 所示。

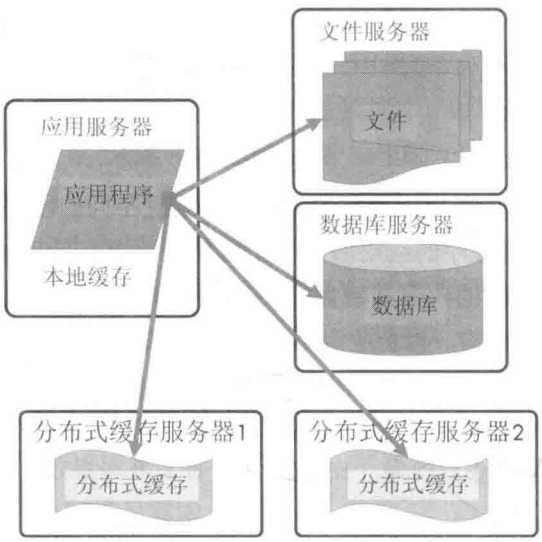


图 1-7　使用了分布式缓存技术的 Web 系统

4. 使用集群技术

应用服务器作为 Web 系统的入口，承担了大量的请求处理工作，随着用户数量的增加和业务范围的扩展，业务用户请求将成几何倍数增长。这种情况下，通常会使用应用服务器集群来分担所有的用户请求。但是这会带来一个问题：用户仅通过同一个地址访问 Web 服务，如何将这样的请求分发到不同的 Web 应用服务器上？分发的原则又是什么？针对该问题，通常的解决方案是在应用服务器集群前面部署一个负载均衡服务器，通过调度来分发用户请求。负载均衡服务器根据分发策略将请求分发到多个应用服务器节点上，如图 1-8 所示。

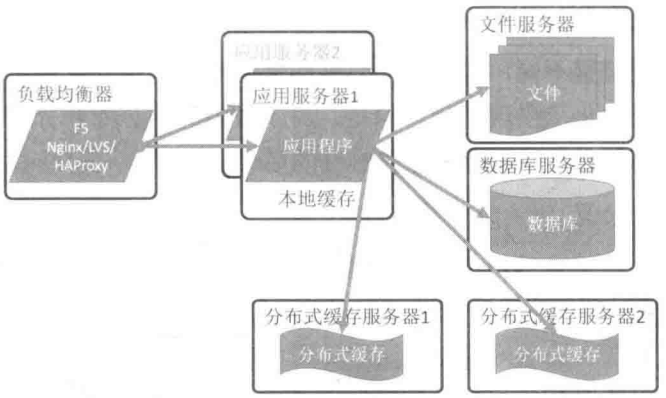


图 1-8　应用服务器集群技术在 Web 系统中的应用

常用的负载均衡技术可分为软件实现和硬件实现两种。典型的硬件方案是 F5，性能和稳定性都很优越，但是价格较为昂贵。软件的解决方案可以采用 LVS、Nginx、HAProxy 等产品。LVS 是四层负载均衡，它根据目标地址和端口选择内部服务器；Nginx 和 HAProxy 是七层负载均衡，它们根据报文内容选择内部服务器。因此 LVS 的分发路径优于 Nginx 和 HAProxy，性能要高些，而 Nginx 和 HAProxy 则更具配置性，如它们可以用来做动静分离(根据请求报文的特征，选择使用静态资源服务器还是应用服务器)。

5．数据库改造

随着用户量的增加，通过上述技术解决了用户请求处理的问题之后，数据库的读写性能将成为实现过程中最大的系统瓶颈，此时也是通过数据库服务器集群的方式来提升整体的性能。改善数据库性能常用的手段是进行读写分离和分库分表。读写分离，就是将数据库操作分为读库和写库，通过主/备功能实现数据同步。如图 1-9 所示，Web 应用程序会将写操作(插入、更新)请求只发给主服务器，而将读操作(查询)请求发送给其他备服务器。这种读写分离技术特别适合那些读多写少的业务应用，如新闻服务。

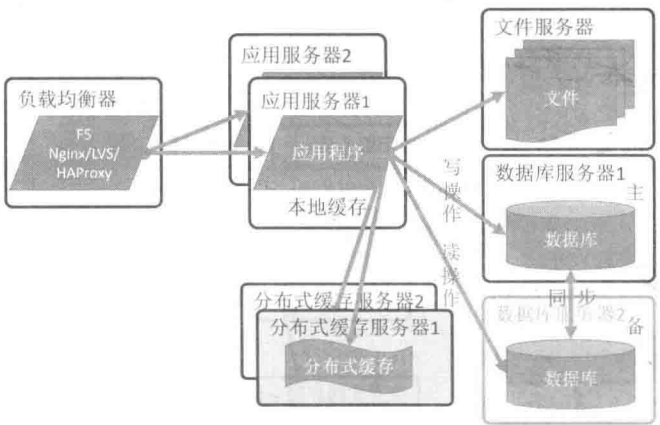


图 1-9　数据库使用主/备方式进行升级

此外，还可以使用分库分表方式，对数据库和表进行水平切分和垂直切分，水平切分是对一个数据库特大的表进行拆分，例如用户表的拆分；垂直切分则是根据业务的不同来

切分，如将用户业务、商品业务相关的表放在各自不同的数据库中。这样一来，由于被操作的数据所在的逻辑位置不同，对于不同数据的读操作和写操作会被分发到不同的数据库服务器上执行，从而避免了出现瓶颈。但是，对于数据库的分库和分表需要资深的数据库管理人员才能完成，稍有不慎将会带来灾难性的后果。

6. CDN 和反向代理

代理(Proxy)是一种特殊的网络服务，通过该服务一个网络终端(一般为客户端)可以与另一个网络终端(一般为服务器)进行非直接的连接。一些网关、路由器及其他网络设备具备网络代理功能。通常，局域网用户无法直接访问广域网服务，需要通过代理服务器间接地访问位于广域网中的服务。此时，局域网用户了解广域网中代理服务器的存在。反向代理(Reverse Proxy)与代理相反，反向代理服务器通常是部署在局域网的机房内，接收来自广域网上的请求，然后将收到的请求转发到局域网的其他服务上，并将其他服务器的处理结果封装后发送给广域网上的用户。此时，广域网的用户并不了解局域网中反向代理服务器的存在。常见的反向代理有 Squid 和 Nginx。

网络上有一句调侃的话："世界上最遥远的距离，不是天涯海角，而是我在电信，你在网通。"这句话生动地描绘了这样一个事实，在互联网世界里，在不同运营商之间进行通信的网络延迟最大。假如 Web 系统部署在成都的机房，那么对于四川的用户来说访问它的速度较快，而对于北京的用户来说访问它的速度相对较慢，这是由于四川属于电信通信发达地区，而北京属于联通通信发达地区，北京用户访问时需要通过互联路由器经过较长的路径才能访问到成都的服务器，返回路径也同样较长，所以数据传输时间比较长。对于这种情况，常常使用 CDN(Content Delivery Network，内容分发网络)来解决，CDN 将数据内容缓存到运营商的机房，用户访问时先从最近的运营商获取数据，这样将大大缩短网络访问的路径，加快访问速度。

通常，Web 系统会将 CDN 和反向代理技术相结合来使用，如图 1-10 所示，反向代理服务器将 CDN 缓存的数据返回给用户，如果没有发现缓存数据才会继续访问应用服务器获取数据，这样做减少了获取数据的成本。

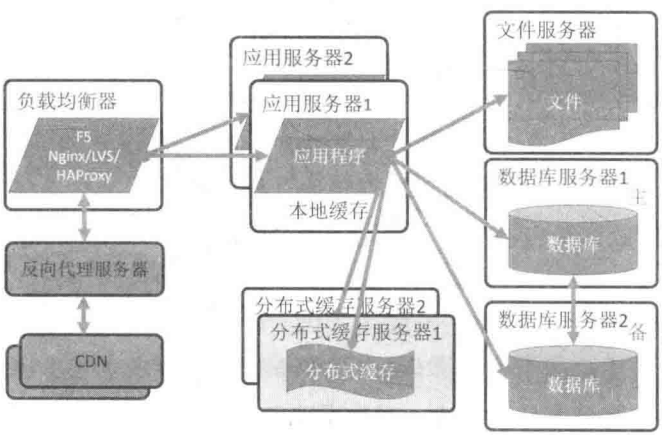


图 1-10　使用 CDN 和反向代理技术