第 3 章　客户—服务器端架构

在第 2 章介绍了分布式系统的定义和软硬件方面的特点。在本章的学习中要理解分布式系统的结构模型。所谓结构模型，是关于其各部分的布局及其之间的相互关系，并定义系统的各组件之间相互交互的方式以及它们映射到下面的计算机网络的方式。分布式系统的组织结构有多种，比较常见的模型是客户—服务器模型(Client/Server Model)，简称 C/S 模型。

3.1　客户—服务器模式的基本概念和优点

3.1.1　客户—服务器模式的基本概念

客户—服务器也可以被理解为是一个物理上分布的逻辑整体，它是由客户机、服务器和连接支持部分组成。客户机是一个面向最终用户的接口设备或应用程序，它是一项服务的消费者，可向其他设备或应用程序提出请求，然后再向用户显示所得信息。服务器是一项服务的提供者，它包含并管理数据库和通信设备，为客户请求过程提供服务；连接支持部分是用来连接客户机与服务器的部分，如网络连接、网络协议、应用接口等。图 3-1 就是一个典型的客户—服务器模型。

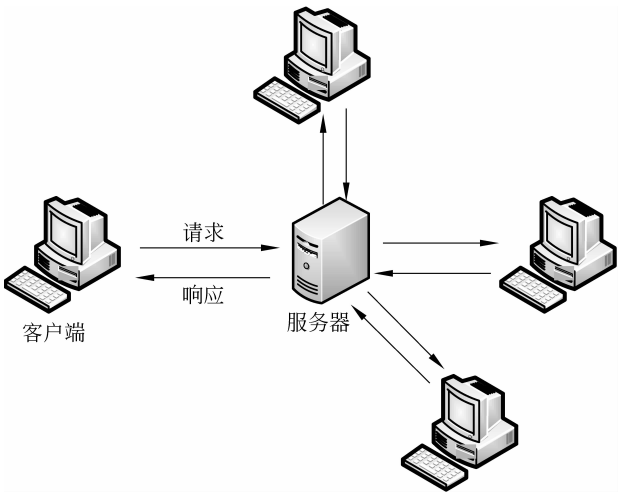


图 3-1　典型的客户—服务器模型

在客户—服务器模型中，服务器是核心，而客户机是基础，客户机依靠服务器获得所需要的资源，而服务器为客户机提供必须的资源。通过它可以充分利用两端硬件环境的优势，将任务合理分配到客户端和服务器端来实现，降低了系统的通信开销。

3.1.2 客户—服务器模式优点

客户—服务器模式主要有以下几方面的优点。

(1) 有利于实现资源共享。网络中的资源具有分布不均匀性，各个不同结点之间的软硬件配置都存在很大差别。在C/S结构中的资源也是分布的，一般来说服务器在软硬件配置上或是数据资源分布上相对客户机而言都具有一定的优势，而且客户机与服务器具有一对多的关系和运行环境。用户不仅可存取在服务器和本地工作站上的资源，还可以享用其他工作站上的资源，实现了资源共享。

(2) 有利于进程通信的同步。分布式系统中的面临的一个重要的问题就是同步问题。在客户—服务器模型中，每一次通信由客户端进程发起请求，而服务器进程一直处于等待状态，以保证及时响应客户端发出的请求。当客户端发出请求后，服务器端响应客户端请求，并以此实现进程间的同步。

(3) 可实现管理科学化和专业化。系统中的资源分布在各服务器和工作站上，可以采用分层管理和专业化管理相结合的方式，用户有权去充分利用本部门、本领域的专业知识来参与管理，使得各级管理更加科学化和专业化。

(4) 可快速进行信息处理。由于在C/S结构中是一种基于点对点的运行环境，当一项任务提出请求处理时，可以在所有可能的服务器间均衡地分布该项任务的负载。这样，在客户端发出的请求可由多个服务器来并行进行处理，为每一项请求提供了极快的响应速度和较高的事务吞吐量。

(5) 具有更好的可扩展性。由于C/S是一种开放式的结构，因此可有效地保护原有的软、硬件资源。以前，在其他环境下积累的数据和软件均可在C/S中通过集成而保留使用，并且可以透明地访问多个异构的数据源和自由地选用不同厂家的数据应用开发工具，具有高度的灵活性；而以前的硬件也可完全继续使用，当在系统中增加硬件资源时，不会减弱系统的能力，同时客户机和服务器均可单独地升级，故具有极好的可扩展性。

3.2 客户—服务器端架构和体系结构

3.2.1 面向连接服务与无连接服务

分布式系统中各主机之间进行通信或数据传输需要计算机网络的通信子网提供的通信服务，通信服务分为两类：面向连接服务和无连接服务[179]。

1. 面向连接的服务

面向连接的服务是指通信的双方在通信过程中必须先建立一个虚拟的通信线路，包括3个过程：数据传输之前先建立连接，在数据流传输过程中维护连接，与数据传输结束后释放连接。例如常见的TCP协议就是一种面向连接服务的协议，客户端与服务器端经

过“三次握手”建立传输连接，即客户机向服务器发出连接请求报文，服务器进程同意建立连接后向客户进程发送应答报文，客户进程接收到服务器进程的应答报文后向服务器进程再次发送建立连接的确认报文。电话系统也是一种面向连接的模式。

由于面向连接的服务在通信的双方之间事先建立了一个连接，因而传输的可靠性好，但是协议复杂，通信效率相对于无连接服务较低，适用于通信不是很可靠的广域网系统中。在许多客户—服务器系统中采用的就是面向连接服务，能够保证通信的稳定和数据传输的正确性。当客户请求服务时，在客户与服务器之间建立一个连接，然后客户再发送一个请求，服务器响应请求，在同一个连接中传回一个应答消息，然后才断开该连接。

2. 无连接的服务

无连接的服务不需要通信双方事先建立一条通信线路，也因此无须经过连接建立、连接维护、释放连接这 3 个步骤，所以无连接的服务通信过程相对简单。无连接服务中每个报文分组都应该保存了完整的目的地址，且每个报文分组都是独立传输的，因而每个报文分组传输的路径可能相同也可能不同，一般是根据当前的网络传输状态为每个报文分组选择路径。例如常见的 IP(互联网协议)、UDP(用户数据报协议)就是一种无连接协议，邮政系统是一种无连接的模式。

不同于面向连接服务事先建立好了一个连接通道，并保证数据传输的可靠性，无连接的服务可靠性并不是很好，但是正是由于节省建立连接的开销，且其通信协议更为简单，因而无连接服务的通信效率更高。

在局域网中，底层的网络相对稳定，客户与服务器之间的通信可以利用开销较小的无连接的服务。如果客户想要发送一个请求时，直接向服务器发送一个请求消息，消息中注明请求的内容和必要的数据，服务器接收到该请求消息后，将处理结果封装在应答消息中，并将该消息送回给客户。表 3-1 是比较了面向连接的服务与无连接服务的优点和缺点。

表 3-1 面向连接服务与无连接服务优缺点比较

服务类型	优　　点	缺　　点
面向连接服务	实时通信 可靠信息流 信息回复确认	占用通信道
无连接服务	不占用通信信道	非实时通信 信息流可能丢失 信息无回复确认

3.2.2 应用程序的层次结构

客户—服务器体系结构是把某项应用或软件系统按逻辑功能划分为客户软件部分和服务器软件部分。客户软件部分一般负责数据的表示和应用，处理用户界面，用以接收用户的数据处理请求并将之转换为对服务器的请求，要求服务器为其提供数据的存储和检

索服务；服务器端软件负责接收客户端软件发来的请求并提供相应服务。

事实上业界对于客户—服务器模型并未提出一个明确的分层结构，但是大多数客户—服务器模型的应用是为用户提供访问数据库的服务，所以可以从用户界面层、逻辑事务处理层、数据层这3个层次进行讨论。

1. 用户界面层

用户界面层是用户通过用户界面中的一些友好提示信息与服务器进行交互的一个层次。其实多数的用户界面层都大同小异，即用户在程序的提示下通过输入设备输入信息，然后客户端将这些信息或数据提交给服务器，交由服务器进行处理。

不同应用程序的用户界面的差异也会很大，主要表现在界面的复杂程序上。如自助银行ATM机的输入设备只能进行简单的数字输入，因而其用户界面也较为简单，一般只需要根据提示输入数字字符。当然也有一些用户界面更为复杂的系统，在这些系统的客户端大多都设计用户友好的图形界面，使用弹出式或者下拉式菜单，而多数是通过鼠标而不再是简单的键盘输入进行操作。

2. 逻辑事务处理层

逻辑事务处理层是在客户端用户提出请求后，服务器对客户端提交的请求服务进行处理，也是整个系统的核心，包括了最主要应用处理程序，在这一层中集中对所有任务进行处理。由于每个系统所实现的功能都不一样，所以系统的处理层所实现的功能也大不相同，主要根据所要完成的任务类型决定。

下面以搜索引擎为例。事实上搜索引擎的用户界面相当简单：即一个简单的提供给用户输入查询信息文本框和一个“确认”按钮。在服务器端存放的是一个巨型的数据库，存放着大量Web页面。搜索引擎的核心是将用户输入信息作为查询的关键字字符串，在后台的数据库中查找相匹配的Web页面，再将查询到的结果组织成一个列表后形成一个新的HTML页面，通过该页面来显示符合要求的结果。在客户—服务器模型中进行信息检索过程一般都是放在逻辑事务处理层来实现的。

3. 数据层

数据层是整个客户—服务器模型的基础，一般都是由服务器提供，它为逻辑事务处理层提供处理过程所需要数据。数据层有时也包含维护用于应用程序操作的数据的程序。数据层的功能包括提供数据存储和数据维护。一般在数据层都会维护一个数据库，用于存储数据。然而数据层除了要存储数据之外，还需要对数据库中的数据进行维护，其中两个重要的方面就是保持数据的一致性和完整性。

数据层为逻辑事务处理层提供服务，在这一层上不需要太关注任务的处理，只需要注重存取数据。值得注意的是在分布式系统中，由于存取的数据不一定都是来自于数据库，也可能是如XML(Extensible Markup Language，可扩展标记语言)、Excel等，而且不同的数据库之间的差别也很大。解决数据的异构性问题的一个方法就是为数据层提供一个接口，逻辑事务处理层只需要调用这些接口，而无须理会数据层读取存在的这些差异以及

数据存取的细节问题，通过这些接口逻辑事务处理层就能够方便的实现异构数据的存取和调用。

3.2.3 客户—服务器模型体系结构

1. 传统的双层体系结构

根据3.2.2节中对应用程序的分层结构，把应用程序的各层上的程序分布到各台计算机上去。为简单起见，先仅考虑两类，即客户机和服务器。客户—服务器模型可能组织结构有如图3-2所示的几种。

图3-2(a)所示的结构表示只在客户机上放置用户界面中与终端有关的部分；图3-2(b)所示的结构表示用户界面程序全部放置在客户端，这种方法就在本质上将应用程序划分为图形前端和后台处理端，前端除了显示应用程序的界面之外，并不进行应用程序的处理工作；图3-2(c)所示的结构是将一部分的应用程序放置到客户机上，客户机执行一小部分的处理的功能，采用这种方式有利于进行一定的用户交互行为。图3-2(d)和图3-2(e)是目前采用的比较多双层体系结构客户—服务器模型。图3-2(d)所示的结构表示将用户界面和应用程序都放置在客户机上，而数据则存放在服务器上，客户机通过对服务器的访问，对数据库进行存取。这样客户机就承担很大一部分工作，而服务器仅需对数据库进行存储和维护工作。图3-2(e)是目前仍然比较流行的一种结构。在客户端设置了数据库，可以缓存一部分的常用的数据，以减少客户机对服务器的访问次数，提高工作效率。

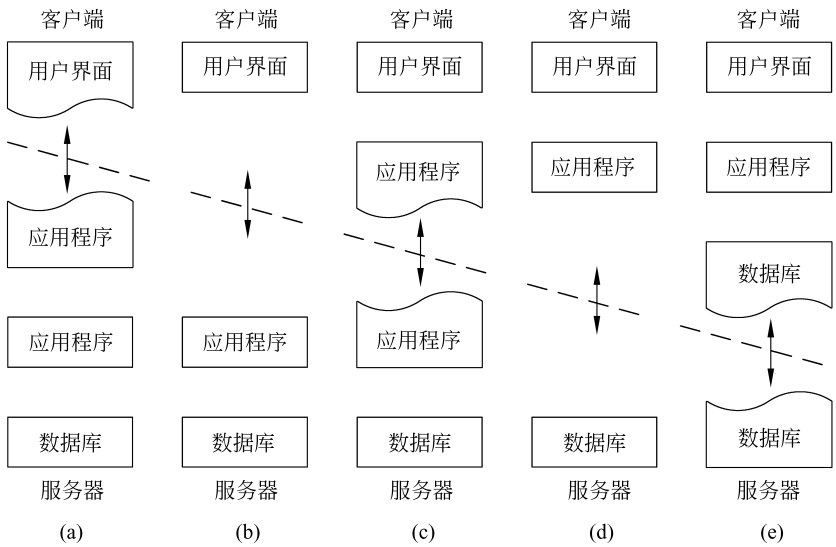


图3-2 客户—服务器模型可能组织结构

传统的客户—服务器模型是双层体系结构，用户界面与逻辑事务驻留在客户机上，而将大部分的数据存放在数据层的数据库中，对数据的操作如查询、修改等由客户机提出请求，数据库存放的服务器将结果返回给客户端，简而言之，客户端整合的应用程序的用户

界面层和逻辑事务处理层两层，而把数据层放在服务器端。这种双层体系结构虽然简单，却有它的一些局限性。图3-3就是双层体系结构的客户—服务器模型。

客户—服务器双层体系结构存在以下几个局限性。

（1）缺乏有效的安全性。由于客户端与服务器端直接相连，当在客户端存取一些敏感数据时，由于用户能够直接访问中心数据库，就可能造成敏感数据的修改或丢失。

（2）客户端负荷过重。随着计算机处理的事务越来越复杂，客户端程序也日渐肥大。同时由于事务处理规则的变化，也需要随时更新客户端程序，就相应地增加了维护困难和工作量。

（3）服务器端工作效率低。由于每个客户端都要直接连接到服务器以访问数据资源，这就使得服务器不得不因为客户端的访问建立连接而消耗大量本就十分紧张的服务器资源，从而造成服务器工作效率不高。

（4）容易造成网络阻塞。正如前面所述，客户端的每次访问都要连接服务器，使得网络流量剧增，容易造成网络的阻塞。

2. 多层体系结构

随着客户机要求处理的事务的数目增多，系统的任务日益繁重，导致系统的吞吐量下降，以及人们对数据安全性有更加强烈的要求。考虑到客户—服务器模型的双层体系结构的种种局限性，人们提出了三层体系结构模式，具体见图3-4。三层体系结构是完全按照应用程序的分层结构来划分的，三层体系结构客户—服务器模型中的三层分别与应用程序的三层相对应。

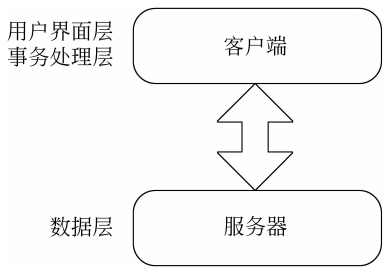


图3-3　双层体系结构的客户—服务器模型

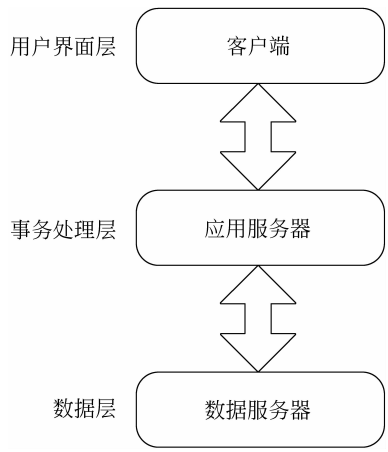


图3-4　三层体体系结构的客户—服务器模型

在该体系结构中，用户界面保存在客户端，事务逻辑保存在应用服务器中，数据保存在数据库服务器中。客户机只负责提供用户界面，当需要进行数据访问时或复杂计算时，客户机向应用服务器发出请求，应用服务器响应客户机的请求，完成复杂的计算或者向数据库服务器发送SQL语句由数据库服务器完成相应的数据操作，最后由应用服务器将结果返回给客户机。需要说明的是，三层体系结构模式的三层是指逻辑上的三层结构（即用

户界面、事务处理层、数据层)而不一定是指物理上的三层结构。

多层体系结构的主要特点。

(1) 安全性。中间层隔离了客户直接对数据服务器的进行访问,从而保护了数据库中数据的安全性。

(2) 稳定性。由于有中间层缓冲客户端与数据库的实际连接,使数据库的实际连接数量远小于客户端的应用数量。

(3) 易维护。由于事务处理层独立于客户端,位于应用服务器中,所以即使事务处理规则发生变化,客户端程序也可以基本不做改动。

(4) 快速响应。通过负载均衡以及中间层缓存数据能力,可以提高对客户端的响应速度。

(5) 系统扩展灵活。基于多层分布体系,当业务增大时,可以在中间层部署更多的应用服务器,提高对客户端的响应,而所有变化对客户端透明。

3. 现代体系结构

前面所述的两种体系结构都是针对于将应用程序划分为用户界面层、事务处理层、数据层分层结构所提出的。各层直接与应用程序的各层逻辑程序相对应,所形成纵向分布多层体系结构,叫做纵向分布结构,相对应的有横向分布结构。与纵向分布结构中将逻辑上不同的组件分别放置在不同计算机上的实现方法不同,横向分布结构是将客户机或者服务器在物理上分割成几个部分,这几个部分在逻辑上拥有同等地位,但是各部分都可以对自己拥有的数据集进行处理,从而使负载得到平衡。事实上横向分布结构更加符合分布式系统的目标。

例如,目前的大型网站访问量都很大,如果只有一台服务器处理客户机的请求,当遭遇访问高峰期时,客户机势必要等待很久。如果有多台服务器,且这些服务器中都保存了完全相同的数据库内容,这样,当一个客户机向服务器请求服务时,服务器端接收到请求,系统会采取循环策略,将服务请求循环转发给其他服务器,且总是由较为空闲的服务器为其提供服务,而不是完全由一台服务器承担所有的任务,这样就使负载得到平衡,如图 3-5 所示。

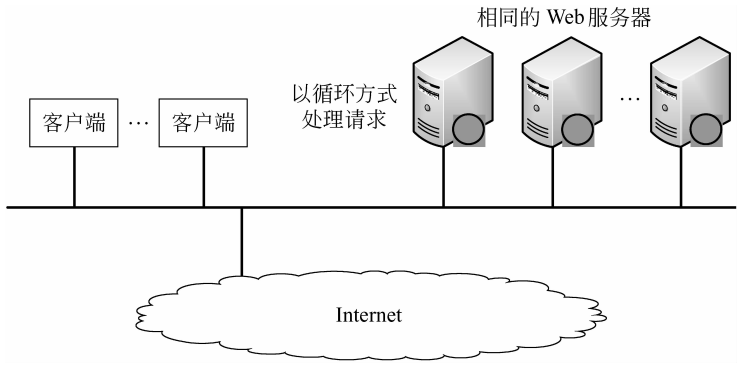


图 3-5 横向分布结构的 Web 服务

除了以上提及的纵向分布结构和横向分布结构，客户—服务器模型还有一些既是纵向分布，也是横向分布的结构。例如前面提到的 Web 服务的例子，还有一些是在客户端和 Web 服务器之间添加了一个用于分配任务的服务器，所有客户端请求的服务先提交到这个用于分配任务的服务器，然后这个服务器根据请求的类型将各类任务分配到各个服务器，这有些类似于前面所提到的客户—服务器模型的三层体系结构。然而在分配了任务的服务器之间也可以将任务负载平衡，横向地分配到多个相同的服务器，这种结构一般用于服务请求较多且较复杂的系统。

3.3 客户—服务器模型的进程通信

计算机网络通信过程实质是分布在不同地理位置的主机进程之间进行通信的过程，进程间的通信实际就是进程之间相互作用，客户—服务器模式实际上就是提供了进程间相互作用的一种方式。在客户—服务器模式中客户与服务器分别表示相互通信的两个应用程序进程，客户向服务器发送服务请求，服务器响应并处理客户的请求，最后返回处理的结果，并为客户提供请求的服务。发起通信、提出请求服务的进程就叫做客户进程，响应请求、提供服务的进程就是服务进程。简而言之，客户—服务器的工作模式是，客户与服务器之间采用，如 TCP/IP、IPX/SPX（Internetwork Packet Exchange/Sequences Packet Exchange，网络分组交换/顺序分组交换）等网络协议进行连接和通信，由客户端向服务器发出请求，服务器端响应请求，并进行相应服务。本节将分别介绍进程通信过程中客户—服务器模型的实现方法，以及客户—服务器模型进行进程通信的各类通信协议。

3.3.1 进程通信中客户—服务器模型的实现方法

在分布式系统的客户—服务器模型中客户进程是随机发起服务请求的，在同一时刻可能有多个客户端向服务器端发起服务请求，这就要求服务器必须要能够处理并发请求。服务器处理并发请求主要实现方法：并发服务器和迭代服务器。

1. 并发服务器

并发服务器的核心是使用一个守护程序；处于后台工作，当条件满足时被激活进行处理。守护程序在随着系统启动而启动，在没有客户的服务请求到达时，并发服务器处于等待状态；一旦客户机的服务请求到达，服务器根据客户的服务请求的类型，去激活相应的子进程，而服务器回到等待状态；并发服务器叫做主服务器，把子服务器叫做从服务器。然后直接由从服务器和客户机进行通信，而不是通过主服务器转发给从服务器，这样就减轻了主服务器的负担。主服务器相当于分配任务的服务器，而从服务器则是真正执行服务请求的服务器。事实上这种方式有些类似于前面提到的客户—服务器模型的横向分布结构的情况。图 3-6 是客户—服务器模型并发服务器的进程通信过程。

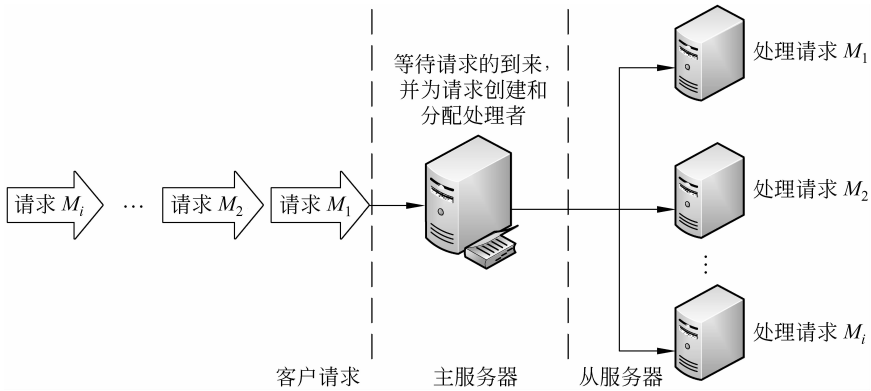


图 3-6　并发服务器进程通信过程

采用并发服务器的客户—服务器模型只要系统允许，可以处理多个客户的服务请求，从服务器是从主服务器接收到任务后，就可以独立的处理客户端的服务请求，并不依赖于主服务器。而且通过采用这种方式不同的从服务器可以分别同时处理不同的客户的服务请求，如果不存在服务器之间交互，则不同的客户服务请求相互之间不会产生影响，正因为如此，采用并发服务的系统具有很好的实时性和灵活性。但并发服务器对系统资源和硬件设备的要求都比较高，一般用于处理不可在预期时间内处理完的服务请求，针对于面向连接的客户—服务器模型。

2. 迭代服务器

通过设置一个请求队列存储多个客户的服务请求，服务器采用先到先服务的原则响应客户端的请求，其他客户则必须在请求队列中等待直到服务器空闲。这种一次只响应一个客户的服务请求，在处理一个请求时其他请求必须等待，迭代响应所有客户端请求的服务器为迭代服务器。图 3-7 是客户—服务器模型迭代服务器的进程通信过程。

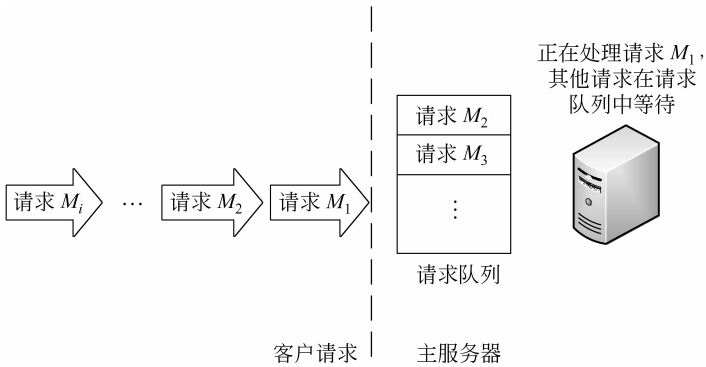


图 3-7　客户—服务器模型迭代服务器的进程通信过程

迭代服务器应用范围并不十分广泛，其主要原因就是服务器对客户服务请求的处理效率低下。迭代服务器常用于提供一些简单的服务，对于较复杂的服务的提供来说，长时

间地停顿在为一个客户的服务上而拒绝其他客户的服务请求是不可取的。迭代服务器相对于并发服务器而言实现的方法更加简单，对系统资源的要求也不高，一般用于处理可在预期时间内处理完的服务请求，针对于面向无连接的客户—服务器模型。而且由于采用迭代服务器方案是将客户端的服务器请求先存放到服务请求队列中，所以迭代服务器处理客户端的服务请求的数量还受到请求队列长度的限制，但是它却能够有效地控制服务请求处理的时间。表3-2比较了并发服务器与迭代服务器各自的特点。

表3-2　并发服务器与迭代服务器的比较

	差　　异
并发服务器	系统资源要求较高 可以处理多个客户的服务请求 从服务器不依赖主服务器而独立处理客户服务请求 不同的从服务器可以分别处理不同的客户的服务请求 系统的实时性好 适应于面向连接的服务类型
迭代服务器	系统资源要求不高 处理客户的服务请求的数量受到请求队列长度的限制 可以有效地控制请求处理的时间 适应于无连接的服务类型

3.3.2　客户—服务器模型的进程通信协议

分布式系统是一个庞大、复杂的系统，要保证整个系统能有条不紊地工作，就必须制定出一系列的通信协议。进程间通信是分布式系统的核心，由于没有共享存储器，分布系统中的通信都是基于底层网络提供的低层消息传递机制的。在本节中对客户—服务器模型进程通信协议的讨论主要是基于OSI参考模型（Open System Interconnection Reference Model，开放系统互连参考模型）而言的。

OSI参考模型是设计用来支持开放式系统间的通信，所谓的“开放式”指只要是遵循OSI标准的系统就可以与位于世界上任何地方、同样遵循同一标准的其他任何开放式系统进行通信。

OSI参考模型是层次结构模型，它将通信过程划分有7层：物理层、数据链路层、网络层、运输层、会话层、表示层（Presentation Layer）和应用层，每一层一个模块，负责处理通信中的某个特定方面的问题，并具有自己的一套通信协议。这种分层的体系结构具有如下特点：上下相邻的层次之间通过接口进行通信；每一层使用下层提供的服务，并为其上一层提供服务；不同结点之间则是通过协议进行通信的。

现在结合OSI的七层参考模型讨论客户—服务器模型各层的进程通信协议。

1. 物理层

物理层是OSI参考模型的最底层，它利用物理传输介质为网络中的各个结点之间激

活、维护和中断物理连接，发送和接收构成网络通信物理表达的信号，在开放式系统的传输介质上传输各种数据的比特流。物理层的数据传输单元是比特。

物理层与网络硬件设备直接相联系，该层可以协调发送与接收网络介质的信号，并确定访问网络特定区域时，必须使用哪种电缆、连接器和网络接口。

物理层一个最主要的功能就是为数据链路层屏蔽网络的底层物理传输介质的差异，使得数据链路层仅需考虑本层的服务与协议，而不需要考虑网络具体使用了哪些传输介质与设备。数据链路实体通过与物理层的接口将数据传送给物理层，然后通过物理层按比特流的顺序将信号传输到另一个数据链路实体。在这个过程中数据链路层并没有考虑到传输介质与设备的差异。

2. 数据链路层

数据链路层位于 OSI 参考模型的物理层和网络层之间，设立数据链路层是为将有差错的物理线路变为对网络层无差错的数据链路，正确传输网络层用户数据，为网络层屏蔽物理层采用的传输技术的差异性。数据链路层所要实现的功能应包括链路管理、帧同步传输、流量控制、差错控制等功能。

在建立了物理线路之后，在进行通信的双方可以传输比特流后才能建立数据链路。数据链路工作过程包括 3 个阶段：建立数据链路、帧传输和释放数据链路。然而在实际的帧传输过程中，由于网络中可能出现的各种问题势必会出现数据传输的误差，为减少这种误差就要求进行差错控制。数据链路层的差错控制机制是在传输的每一帧的后面加入检错码和纠错码。常用的检错码主要有奇偶校验码和循环冗余编码。当帧送到接收端时，接收端根据约定好的规则再次计算检验码(Check Code)，与传送的帧中附加的检验码进行比较，相同的话就表明帧传输无错，否则表示出错，出错则需要采用反馈重发机制重新发送该帧。反馈重发纠错实现的方法有两种：停止等待方式和连续工作方式。在停止等待方式中，发送方在发送一个数据帧后，需要等待接收方的应答帧的到来，如果应答帧表示上一帧已正确接收，发送方才发送下一帧；否则，重新发送出错的帧。连续工作方式是人们为克服停止等待方式系统通信效率低的原因而引入的，它又分为拉回方式和选择重发方式。拉回方式可以连续地向接收方发送数据，接收方接收数据并对其进行检验，然后向发送方返回应答帧。当发送方接收到接收方数据帧发送出错的应答帧后，就停止发送当前帧，然后重新发送包括出错帧在内以后所有帧。而选择重发方式只发送出错的帧，并不是发出错帧以后所有的帧，效率显然要高于拉回方式。

3. 网络层

网络层位于 OSI 参考模型的第 3 层，该层最主要的任务就是通过路由选择算法，为分组通过互联网络选择适当的路径。网络层要实现路由选择、拥塞控制与网络互连等功能。网络层使用数据链路层提供的服务，并向运输层提供端到端的服务。

消息从发送者传送到接收者的过程中，可能要经过好几次的转发，每次转发都要选择一条线路传送出去，如何选择最佳路径就是路由选择的问题，同时这也是网络层最首要的任务。不同的规模的网络需要选择不同的路由协议，路由选择协议分为两大类：内部网

关协议(Interior Gateway Protocol，IGP)和外部网关协议(External Gateway Protocol，EGP)。内部网关协议是在一个自治系统内部使用的路由协议这与其他自治系统选用什么路由协议无关。当两个使用不同内部网关协议的自治系统进行通信时，就需要使用外部网关协议。目前流行的路由协议主要有3种：路由信息协议(Routing Information Protocol，RIP)与开放最短路径优先(Open Shortest Path First，OSPF)协议是属于内部路由协议，外部路由选择协议有边界网关协议(Border Gateway Protocol，BGP)。

网络层除了路由选择协议外还有一个十分重要的协议，就是IP协议。IP协议具有以下特点。

① IP协议是一种不可靠、无连接的协议。IP协议提供一种无连接的数据报传送服务，它不提供差错校验。无连接表明IP协议并不维护IP数据报发放后的任何状态信息，每个数据报的处理都是独立的。不可靠意味着发送的数据报不一定能成功到达目的结点。

② IP协议是点对点的网络层的通信协议。IP数据报的交付手段可以分为直接交付和间接交付两类。具体采用的类型要根据数据报的目的IP地址和源IP地址是否属于同一个网络来判断，属于同一个网络的机器之间采用直接交付的手段。而采用间接交付手段时，两实体间连接是在同一个网络的路由器—路由器的网络层之间进行的。IP协议是针对两个点对点的通信实体对应的网络层之间的通信协议。

③ IP协议向传输层屏蔽了网络底层的差异。由于网络的异构性，底层网络协议在各个方面都存在很大差异，如帧格式、地址格式的差异等等。通过IP协议，网络层向运输层提供统一的IP数据报，以屏蔽网络在帧结构和地址的差异，事实上这也是网络层一个十分重要的任务。

4. 运输层

运输层位于OSI参考模型的第4层，该层的主要功能是向用户提供可靠的端到端的服务，其主要任务就是实现分布式进程的通信，是整个协议结构的核心。运输层向高层屏蔽了下层数据通信的细节，它是系统通信中关键的一层，为实现应用层的功能提供服务。

传输控制协议(Transmission Control Protocol，TCP)以及用户数据报协议(User Datagram Protocol，UDP)与IP协议结合而成的TCP/IP协议是目前网络通信中实际上用到的标准。运输层协议需要具有从创建进程到进程通信在各运输层提供流量控制机制的功能。TCP/IP协议族为运输层设计了两个协议：UDP协议和TCP协议。其中，UDP协议是一种无连接的传输层协议，它是在一个低水平的基础上完成传输所要求的功能；TCP是一种面向连接的传输层协议。

分布式系统中，客户—服务器模型中的交互通常是利用运输层的协议进行，客户—服务器模型的应用程序和系统也通常是利用TCP协议构成的。TCP能够提供可靠的服务，但是开销也相对地要增加，对于相对稳定可靠的底层网络，如果仍采用TCP协议构建客户—服务器模型其性能和效率也相对较低。一种较好解决方案就是采用UDP，并对特定的应用程序提供附加的差错控制和流量控制手段。

5. 会话层

会话层(Session Layer)位于OSI参考模型的第5层,该层的主要功能是负责维护两个结点之间会话连接的建立、管理和终止,以及数据的交换。

6. 表示层

表示层位于OSI参考模型的第6层,该层的主要功能是处理两个通信系统中交换信息的表示方式,包括数据格式变换、数据加密和解密、数据压缩与恢复功能。

7. 应用层

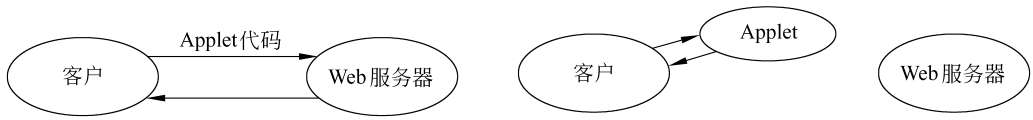
应用层(Application Layer)是OSI参考模型的最高层,该层的主要功能是为应用程序提供网络服务。应用层需要识别并保证通信双方的可用性,使得协同工作的应用程序之间的同步,建立传输错误纠正和保证数据完整性控制机制。

3.4 客户—服务器端模型的变种[71]

客户—服务器端模型是目前流行的一种体系结构模型,重点考虑某些方面的因素,客户—服务器模型能派生出几个变种,如考虑到网络通信问题,派生出的移动代码和移动代理;考虑到用户需要硬件资源有限的、基于管理的低价格的计算机而派生出的网络计算机和瘦客户;另外为了能以方便的方式建立和删除移动设备,派生出了移动设备和自组网络等。

3.4.1 移动代码

移动代码是指能从一台计算机上下载到另一台计算机运行的代码。例如,Applet(小移动应用程序)是一种众所周知和广泛使用移动代码:用户运行浏览器选择一个到Applet的链接,Applet的代码存储在Web服务器上,代码被下载到浏览器并在那儿运行,如图3-8所示。



(a) 客户请求下载Applet代码　　(b) 客户与Applet交互

图3-8 Web Applet应用程序

本地运行下载的代码的好处是,由于不会遭遇跟网络通信相关的延迟或带宽的变化(可变性),因此会得到很好的交互式响应。

但是移动代码对于目的计算机里的本地资源有潜在的安全威胁,恶意的移动代码可能会降低系统的安全性能,对本地计算机造成安全隐患,如一些蠕虫病毒和木马程序,而

浏览器也会因此而限制 Applet 对本地资源的访问。

3.4.2 移动代理

移动代理是可以从一台计算机移动到网络上另一台计算机，访问本地计算机的资源，完成存储信息收集之类的任务，最后返回结果的一种程序，它包括代码和数据两部分。移动代理能在异构计算机网络中的主机间自主的迁移的程序，它在汲取传统分布计算技术的有益经验的基础上，为分布计算提供了一个全新的范型。移动代理的代理实体的运行不是固定在一台计算机，而是可以在多台计算机上。移动代理系统的例子有：IBM 的 Aglet 以及 Voyager，AgentTCL 等。

一般情况下，移动代理运行在特定的虚拟机（Virtual Machine，VM）环境中。移动代理与普通程序的最大不同就是它可以在运行期间在虚拟机之间迁移而无须中止程序的执行。移动代理的特点在于移动上，它可以选择何时进行迁移，移动到何地。主要表现在每个代理可以在执行的任意点上挂起并将自己传送到另一台主机上，然后在该处继续执行，任务结束后将执行结果返回给原主机。移动代理系统具有跨平台的特性，其工作平台可以混合使用不同厂家的系统。通过在各个操作系统中运行移动代理的虚拟机，可以将硬件和操作系统的平台细节屏蔽，使代理获得一个统一的界面。在这个基础上，代理可以在各个平台之间自由移动。它还可以执行克隆等操作，产生子代理共同完成任务。与传统的代理相比，移动代理具有以下特点：

（1）主机间动态迁移。不同于传统构架中代理固定在特定的主机上，移动代理则可以在运行期间直接进行主机间迁移。传统的代理收集到的数据发送给上级处理器或是其他代理，而移动代理则一台计算机上采集所需数据并进行处理后，在终止进程的情况下直接迁移到另一台主机运行，并保留了原进程数据。这样相对于传统代理需要进行一个较复杂的通信过程，移动代理简化了数据的处理过程，从根本上改变了数据的可操作性和全局性。

（2）智能性。由于移动代理可以自由地在主机之间进行迁移，使得代理的运行场地不再局限在某一个特定位置，从而比较容易获得全面和有针对性的数据。在这些数据的基础上，代理可以充分利用现有的人工智能和统计技术，做出更加及时和准确的判断。这种特性使得移动代理与传统代理相比，可以更有效地自主完成某一个特定的任务。

（3）平台无关性。多数移动代理采用与平台无关的语言，这样的程序可以跨平台运行。由于主流的平台无关语言（例如 Java）在各种操作系统上都有其相应的实现，所以选用这些语言的移动代理可以很容易地完成跨平台的连接。另外，一般的移动代理体系都建立了与移动代理相配套的平台无关的通信协议。通过这些协议，代理之间无须建立直接的通信连接，而是利用虚拟机提供相应的消息服务，简化消息传递的操作。在这个基础上，可以更容易地开发异构平台上的应用系统。

（4）分布的灵活性。移动代理运行在整个分布式系统中，而不是固定在某一个特定的位置。这样，一旦需要，它可以将自己或者所需的其他移动代理直接发送到所需的主机

现场，进行本地操作。这样，提高了操作的灵活性，同时也消除了传统代理间通信时对复杂通信协议的依赖性。

（5）低网络数据流量。由于结构上的特殊性，移动代理可以实时对所采集到的数据进行过滤，然后将关键数据提出，而无须像传统的代理体系那样，将各个主机的所有数据都汇集到一个中央服务器中，由这个服务器进行综合处理，然后再向相关的代理转发。这样，可以明显减少经过网络上的数据流量，提高网络的总体可用性。

（6）多代理合作。多代理合作是移动代理的一个重要特性。也就是说，通过虚拟机系统的通信机制，可以实现多个代理之间的合作。这种合作有多种模式。相同的代理之间互相协作，可以防止系统和代理失效。一旦有代理失效，其他代理可以采取措施，通过承担起失效代理的任务或者启动新的代理的办法来进行失效弥补。另外，异种代理之间也可以进行互补性合作，多个不同功能的代理协作完成共同目标。这样，有利于将总体功能模块化，减少单个代理所完成的功能，从而降低代码的复杂度，缩短调试过程。利用这个特性，可以进一步增强代理的可靠性。

移动代理存在的一个重要问题就是安全问题。每个代理都是一段计算机程序，根据移动的特点，代理可以在网络间传输。每个代理都执行一定的操作，它可以在一台主机上生成；可以被派发到一台远程主机，继续执行自己的操作，并将执行结果传回原主机；还可以克隆，消除等。移动代理对所访问的计算机上的资源存在潜在安全威胁，从某种意义上讲，代理类似于一种病毒。有一些恶意的代理可以获得对远程主机上的文件系统、磁盘、CPU、内存等的控制，进而对对方主机进行破坏。因此每个移动代理系统都要对从远方过来的代理进行身份验证，其环境应该根据代理当前代表用户的身份决定哪些本地资源被允许使用，保护本地资源不被损坏；同时每个代理还要对目的地的真伪进行判断，以防止有人冒充而获取代理携带的信息。

3.4.3 网络计算机

网络计算机是一种专用于网络计算环境下的终端设备。与 PC 相比一般没有硬盘、软驱、光驱等存储设备。它通过网络获取资源，应用软件和数据也都存放在服务器上。有些网络计算机可能也会包含一个磁盘，但是它仅存储少量的软件，其余空间则用做缓存，保存近期从服务器下载的软件和数据文件。

在桌面计算机环境中，操作系统和应用软件需要大量的活动代码和数据，它们位于在本地磁盘上。但是应用文件的管理和本地软件库的维护都需要相当的技术工作，这对于大多数的用户是不能胜任的。网络计算机是对这个问题的一个应对。它按用户的需要从远程文件服务器下载操作系统和任何应用软件。应用在本地运行，但文件由远程文件服务器管理。由于所有的应用数据和代码都由文件服务器存储，用户可以从一台网络计算机迁移到另一台。网络计算机的处理器和存储器能力可以被限制以降低它的成本。

网络计算机的工作原理是，终端和服务器通过 TCP/IP 协议和标准的局域网联结，网络计算机作为客户端将其鼠标、键盘的输入传递到终端服务器处理，服务器再把处理结果

传递回客户端显示。众多的客户端可以同时登录到服务器上，仿佛同时在服务器上工作一样，它们之间的工作是相互隔离的。

3.4.4 瘦客户

瘦客户是指一个软件层，它支持用户端的计算机上基于窗口的用户界面，而在远程的计算机上执行应用程序。这种结构与网络计算机模式有同样低的管理和硬件费用。但它不是下载应用代码到用户的计算机，而是在计算服务器上运行它们。

由瘦客户而引发一种新的产品就是瘦客户机，瘦客户机其实是网络计算机的新一代产品，它是一种应用于网络计算环境的瘦客户机产品，通过网络获取资源，大部分计算和存储都在服务器，适于集中式的应用。瘦客户机有人机交互必需的显示器和输入设备，一般没有硬盘、软盘和光驱等外部存储设备，是一种无噪声、微型、高性价比的网络接入设备。

但是瘦客户体系结构也有其局限性，主要原因是在交互性高的图形活动和图像处理中，用户将感受到包括网络和操作系统的延迟在内的通信延迟，更由于瘦客户端和服务器应用程序进程之间传输图像和向量信息而加大通信延迟。

3.4.5 移动设备和自组网络

现在各种小的、便携的计算设备越来越多，包括笔记本电脑、个人数字助理（PDAs）类的手持设备，移动电话和数字照相机、可穿戴的计算机（例如 Smart Watch）以及嵌入在日常装置（如洗衣机）里的设备。许多的这些设备能够无线联网，范围从大城市或更大的范围（GSM①、CDPD②）到数百米（例如无线局域网 WaveLAN）或几米（例如蓝牙、红外线和 HomeRF）。这些设备提供了对移动计算（Mobile Computing）的支持，从而用户能够在网络环境之间携带它们的移动设备，并利用本地和远程的服务。

集成移动设备和其他设备到一个给定的网络的分布形式可以用自组网络（Spontaneous Networking）来描述。这个术语用于指这样的应用：它涉及以一种比现在可能的更加不正式的方式把移动和非移动设备连接成网络。那些嵌入式设备向用户和其他的相邻设备提供服务。像 PDAs 这样的便携式设备让用户可以访问在他们当前位置的服务，也可以访问像 Web 这样的全球服务。

自组网络的关键特征如下。

（1）易于连接到本地网络。由于是无线连接，因而避免了预装电缆和安装插头、插座的麻烦。另外，在自组网络中，当设备移动到一个新的网络环境中，该移动设备能透明地重配置以获得连接，而用户也不必输入本地服务的名称或地址才能获得本地服务。

---

① GSM（Global System for Mobile Communications，全球移动通信系统）。

② CDPD（Cellular Digital Packet Data，蜂窝数字式分组数据交换网络）。

(2) 易于跟本地服务集成。当移动设备移动到某一设备网络中时,用户不需要进行特殊的配置就能自动地发现该网络提供的服务。

(3) 有限的连接。用户在移动时,由于各种网络状况,并不能保持设备的持续地连接。例如在某些信号的盲区,无线网的连接可能时断时续,就可能使得无线连接中断。

(4) 安全性和隐私性。正是由于自组网络连接的自主特性从而又引起了它的安全问题。例如用户在移动时访问企业内部网的设施可能会暴露在企业内部网防火墙后的数据,也可能打开企业内部网而受到外部攻击。

3.5 小结

客户—服务器模型是分布式系统设计中一种流行的体系结构。事实上目前有许多服务都是基于客户—服务器模型的,如 FTP(File Transfer Protocol,文件传输协议)服务、新闻组和邮件服务、DNS(Domain Name System,域名系统)服务等。本节先从客户—服务器模型的基本概念开始介绍客户服务器模型,并集中讨论了客户—服务器模型相对其他模型的一些优点,如有利于实现资源共享、有利于进程通信的同步、可实现管理科学化和专业化、可快速进行信息处理、具有更好的可扩充性等。

在网络服务中所提供的通信服务有面向连接和无连接两类,在分布式系统中各主机之间进行通信或数据传输需要计算机网络的通信子网提供的通信服务也分为这两类,在本章中详细说明这两类服务实现的原理和它们之间的差异。在本书中将应用程序分为用户界面层、逻辑事务处理层和数据层三层,并分别介绍了这三层所实现的职能,然后根据应用程序的层次结构说明客户—服务器模型的双层体系结构和多层体系结构,多层体系结构是在传统的双层体系结构上进行扩展而得到的。同时本书中也扩展性地说明现代体系结构。

网络本质是分布在不同地理位置的主机之间的进程的通信,本章节说明了客户—服务器模型进程通信的两种具体的实现方法:并发服务器和迭代服务器,并对这两种方法进行了比较。然后根据 OSI 参考模型对进程通信协议进行了阐述,说明各层的功能和任务。

最后,在着重考虑客户—服务器模型中的一些因素,某些更动态的系统可以构造为客户—服务器模型的变种:从一个进程到另一个进程移动代码的可能性允许一个进程委托任务到另一个进程。例如,客户可以从服务器下载代码在本地运行它。对象和存取它们的代码能够被移动以减少访问延迟和最小化通信量,如移动代码和移动代理(Mobile Computing)。某些分布式系统被设计以使计算机和其他移动设备能无缝地添加或删除,允许它们发现可用的服务并向其他设备提供它们的服务,如移动设备组成的自组网络。在一个计算机网络中实际的放置(布局、分布)组成分布式系统的进程可能受到性能、可靠性、安全性和费用的影响,如网络计算机和瘦客户的方法。

习题

1. 什么是客户—服务器体系结构？
2. 客户—服务器体系结构有什么特点？
3. 采用客户—服务器模型有什么好处？
4. 试比较面向连接服务与无连接服务的异同点。
5. 比较两层客户—服务器结构和三层客户—服务器结构的特点。
6. 人们为什么要提出多层体系结构的概念？
7. 试简述 OSI 参考模型的七层模型。
8. 试比较并发服务器与迭代服务器之间的差异。
9. 客户—服务器模型有哪些变种？试举例。
10. 试比较移动代码和移动代理的区别。

第二篇　云计算技术

通过第一篇的学习，已经对分布式计算的概念有了一定的了解。云计算可以看作是一种新兴的分布式计算技术，作为一项有望大幅降低中小企业计算成本的新兴技术，云计算有着广阔的应用前景。本篇我们将向读者详细介绍云计算的相关概念，发展中所面临的机遇和挑战以及各种云平台。

本书第4章将回顾云计算的发展历史，读者可以看到云计算并不是一个全新的事物，它是分布计算，集群计算、网格计算、公用计算等各种技术发展融合的产物。本章将详细地比较云计算与各种已有计算技术的区别与联系；第5章，将从文件系统、数据存储、分布式计算实现技术3个方面解析Google目前的云计算架构。除了Google这些为人所熟知的云计算技术外，Yahoo！公司在云计算方面也投入了大量资源进行研发；第6章将介绍Yahoo！公司的PUNTS、Pig以及ZooKeeper技术。接下来的两章我们将对Greenplum数据引擎以及亚马逊公司的Dynamo数据存储技术进行介绍。新一代数据处理引擎技术的领先者Greenplum公司研发的旗舰产品Greenplum数据引擎，对于下一代的数据仓库和大规模的分析处理，有其独特优势。它已经被福克斯互动媒体、纳斯达克、纽约证券交易等大型企业采用；第7章讲述在云计算环境中Greenplum占据的优势，另一个比较典型的云端数据存储技术是亚马逊公司的Dynamo；第8章将详细讲解Dynamo在开发时所考虑的各方面因素，包括体系结构、应用环境、故障恢复等技术，最后，作为IT行业“蓝色巨人”的IBM公司凭借其在硬件软件上的优势提出了自己的蓝云计划。第9章将对IBM的云计划特别是其虚拟化的动态基础架构技术进行详细描述。

通过本篇的学习，读者将会更加深刻了解云计算的概念，以及云计算技术在发展中要面临的各方面的挑战。云系统的安全性、可扩展性、可用性，数据的存储、故障恢复、虚拟化技术等都是需要考虑优化的问题。相信随着行业巨头的竞争，云计算技术必将得到飞速的发展，扭转整个行业的面貌，为企业和用户使用计算资源带来便利。