PowerBuilder 8.0 网络技术与应用

明德祥　于世江　张智永　等编

国防科技大学出版社

PowerBuilder 8.0 网络技术与应用

明德祥　于世江　张智永　等编

国防科技大学出版社

内 容 简 介

PowerBuilder8.0 可以用来设计、建立高性能的基于网络数据库应用系统的设计开发工具，同时全面支持分布式、客户/服务器体系结构、分布式计算和 Web 环境。

本书介绍了 PowerBuilder8.0 的网络应用技术，讲述了开发分布式应用和 Web 应用的步骤、各种技术、各种工具及其使用方法。全书共分为七章，分别介绍了 PowerBuilder 环境下的网络技术、创建组件、创建客户、在 PowerBuilder 客户中使用 SSL、分布式应用的故障诊断、分布式应用系统实例以及 PowerBuilder 的 Web 应用技术等内容，书中的实例不仅有助于读者理解相关的知识点，而且在实际应用中具有较大的可移植性和可参考性。

本书适合于想了解和使用 PowerBuilder8.0 网络应用技术的 PowerBuilder 用户入门学习使用，同时可作为进行 PowerBuilder 网络应用设计开发的参考书。

图书在版编目（CIP）数据

PowerBuilder8.0 网络技术与应用 / 明德祥　于世江　张智永 等编　　国防科技大学出版社，2002.1
ISBN 7-81024-796-4/TP・178

Ⅰ. P…　Ⅱ. ①明…②于…③张
Ⅲ. 软件工具 ②PowerBuilder8.0
Ⅳ. TP311.56

中国版本图书馆 CIP 数据核字（2001）第 083657 号

PowerBuilder8.0 网络技术与应用

明德祥　于世江　张智永 等编

国防科技大学出版社

电话：0731-4572640　邮政编码：410073

E-mail：gfkdcbs@public.cs.hn.cn

策　　划：陆魁玉　卢天贶

责任编辑：邱建雄　卢天贶

各地新华书店经销

湖南省地质测绘印刷厂印刷

*

787×1092 毫米　16 开本　17.75 印张　427 千字

2002 年 1 月第 1 版　　2002 年 1 月第 1 次印刷

定价：25.00 元

编 委 会

前　言

Sybase 的 EAStudio 是一整套为用户提供全面的企业级 Web、分布式和客户/服务器应用解决方案的应用开发和提交工具包。它包含 4GL 快速开发工具 PowerBuilder8.0、Java 开发工具 PowerJ、集成了动态页面 Web 服务器和组件事务服务器的新一代企业应用服务器 Sybase Enterprise Application Server 等。无论用户是需要开发复杂的 4GL 应用或 Java 程序，还是需要一个适应各种提交需求的中间层体系结构，Enterprise Application Studio 都可以满足用户的要求，用以建立基于 Web 的、分布式的或 Client/Server 结构的企业级应用系统。企业级的开发和提交（Enterprise Application Development and Deployment）是 Sybase 公司的产品策略之一，它包括用户在企业级应用开发过程中一个功能广泛的开发工具集和一个多层应用的提交环境，构成了 Sybase 在这一领域的一个完整的解决方案——EAStudio。

Sybase Enterprise Application Server™是业界第一个将 Web OLTP 和动态信息发布紧密集成的应用服务器，并且支持所有标准的组件模型，这对充分利用原有技术和投资提供了最大程度的灵活性。Enterprise Application Server 集成了操作方便且功能强大的 Web 应用服务器 PowerDynamo，以及高性能的组件事务处理服务器 Jaguar CTS。它可以建立以内容为导向的应用系统，前端通过浏览器展示数据，后端不仅可以提供动态页面，而且具有强大的事务处理能力。企业可以完全在一个完整开发环境中实现基于 Web 的事务处理应用的一切功能。Enterprise Application Server 使展示逻辑部分和事务处理逻辑部分容易地分离，优化了应用开发，能够实现一个易于管理的、易于数据更新的、可伸缩的、可靠的、安全的、高速的企业应用。

PowerBuilder8.0 是 PowerBuilder 的最新版本。它专门用来设计、建立高性能的基于分布式、客户/服务器体系结构的网络应用系统，同时全面支持分布式计算和 Web 环境。PowerBuilder8.0 集成了 PowerSite 功能，利用它可以建立、管理和提交 Web 应用：使用 DataWindow 技术，开发人员可以利用现有的 PowerBuilder 逻辑，把它们重新提交到瘦客户端或 HTML JavaScript 客户端；容易地集成标准的 Web 组件和业务逻辑；它与 Enterprise Application Server 紧密集成，便于开发、管理、调试和提交。PowerBuilder8.0 为建立企业 Web 应用提供了一个全面的、集成的 Web 开发环境。

本书共分为七章，分别介绍了 PowerBuilder 环境下的网络技术、创建组件、创建客户、在 PowerBuilder 客户中使用 SSL、分布式应用的故障诊断、分布式应用系统实例以及 PowerBuilder 的 Web 应用技术等内容，书中的实例不仅有助于读者理解相关的知识点，而且在实际应用中具有较大的可移植性和可参考性。

本书适合于具有一定的 PowerBuilder 使用经验，希望了解或者使用 PowerBuilder8.0 进行网络应用程序开发的读者。通过大量的实例向读者详细介绍 PowerBuilder8.0 的开发网络应用程序的基本原理和各种基本实用的编程技术。不仅可以学会使用开发基本的基于 Jaguar CTS 的分布式应用程序，而且可以学会开发 Web 应用程序的各种实用技术。

由于时间关系和编者水平所限，书中难免有所纰漏，欢迎广大读者朋友批评指正！

编　者

目 录

近 20 年来，信息技术行业中最显著的变化莫过于大型机在时代舞台上逐渐隐去，而由各种网络工作站取而代之。在这个变化中，终端用户获得了比以前更为强大的处理能力，分布于整个企业各处的硬件资源也拥有了比以前更强大的功能。数据中心和无尘微机室一去不复返了，取而代之的是桌面计算机、工作组服务器以及小型机。这种变化最初是从硬件上开始的，而目前则更多地体现在软件方面。所以，我们现在的任务是，开发更适合这些分布式硬件资源发挥潜力的软件环境。

本章介绍在 PowerBuilder 环境下网络技术中分布式应用技术的一般概念，其中包括采用分布式应用的必要性，分布式应用的体系结构以及如何设计一个分布式应用。

需要说明的是，虽然可以在 PowerBuilder 企业版、专业版或桌面版三种产品的任一环境下开发分布式应用中的客户机和服务器应用程序，但只有企业版才提供了完全的分布式计算能力，因此，不能在专业版或是桌面版下运行服务器应用程序。

1.1 分布式计算概述

1.1.1 关于分布式计算

分布式计算，英文为 Distributed Computing。分布式计算系统由集中式计算系统和客户机/服务器计算系统发展演化而来。分布式计算系统基本上是客户机/服务器计算系统的规模扩展。如果我们说某项工作是分布式的，那么，参与这项工作的一定不只是一台计算机，而是一个计算机网络。我们的计算机具有两种功能，一是它能够存贮信息，或者称为数据；另一个功能是它能够处理数据，即能够计算。按这一分类原则，我们可以把计算机网络所做的工作分两种：分布式数据存贮，分布式计算。

在分布式数据存贮中，网络使数据存贮分布化，我们把数据放到网络上的不同的机器中，而不是仅存储在一台计算机里。数据是共享的，网络中的任何计算机可以透明地存取到不同来源的数据。比如，在 PowerBuilder 中，程序所处理的数据往往来自于不同的数据库服务器，而不仅仅是本地机器，因此，我们把它称为分布式数据。

在分布式计算中，网络侧重于它的计算功能。在分布式数据中，完成一件工作时，数据可能来自于网络中不同的机器，但对于这些数据的处理却是在本机中完成的。而在分布式计算环境中，数据的处理不只是在一台机器完成，而是多台机器协作完成。比如，为了处理一项工作 P，它由 PA 和 PB 两部分工作组成。如果我们把 PA 放在机器 A 中完成，PB 放在机器 B 中完成，那么它们就形成了一个分布式的计算。计算机的计算总是离不开数据，所以，在大部分情况下，分布式计算总是伴随着分布式数据，分布式计算往往是一个表示程度的量词。

采用分布式计算有以下一些优点：

（1）共享计算资源

用整个网络中的所有计算机来处理总比单台机器要快一些，另外，一些运算速度比较慢的客户机也可以用运算速度比较快的服务器来协作完成某项工作。

（2）减轻网络负荷

虽然网络技术飞速发展，网络的带宽不断的增加，但这种增加总是有限的。在分布式数据处理中，在网络中传递的数据量是非常大的。SQL 语句会把一整张表放在网络上。因此，如果在网络中所传输的仅仅是一些数据处理后的结果，而不是前面所说的大量的中间数据，整个网络的负荷就会降低了。

（3）增加安全性保障

我们可以把一些关键的计算过程和数据放在服务器上，并给予特殊的安全保护。

（4）软件结构合理化

在某些环境下，拥有分布式计算的软件结构更加容易解决实际问题。

目前，拥有分布式计算资源的计算机网络已经十分普遍，那么，在多种资源间进行的分布式相关的处理不仅具有现实意义，而且还产生了比较急迫的要求。数年以来，针对分布式处理，人们研制出了多种处理机制并在实践上加以一定的运用，其中包括简单纯粹的数据共享到复杂的多层次服务支持系统。下面将对各种分布式计算的方法做一个简要介绍，其中包含了其核心概念原理以及常见的具体实施方法。

1．网络通讯

所有分布式计算环境的基础是计算机之间的通讯。虽然这个过程是最基本的过程，但也是必须的，并且从概念上反映了分布式环境和底层通讯模块的接近程度。我们知道，让计算机和其他计算机进行通讯的硬件以及系统级软件常常称作传输层。而当几个计算机共同使用的传输层相连时，我们就可以称其为计算机网络。

网络上的信息传递过程和我们平时所使用的邮政信件传送过程是十分类似的。就象一个邮包一样，网络上的信息也被打包，包中含有收信者和发信者的地址，以及一些真正需要传送的自带信息，比如一条短消息等等，这些信息通过一些具有邮发功能的机器进行传递。另外，和邮政信件一样，收到网络信息包的人可以选择接收信息包，也可以不接收。

不过，不管是普通信件，或者是网络信息，如果超过了一定的大小限制，那么他们就可能会被分割成多个小部分，等到他们到达目的地的时候才再组合起来。这些从物理上分割的信息包，其实也可以被看作是具有独立逻辑的信息包。一般来说，只要传输层中具有一定的语义、分组顺序、数据格式化和一系列其他预定义的组件，就可以组成某种通讯协议。只要遵循这些预定义的协议，某一计算机系统就能够正确解释来自其他计算机系统的信息。

2．同步和异步传输

和普通的邮件相同，信息发送者关注信息接收者接收信息的情况，其侧重点也各有不同。有时候，也许发送者根本就不需要关心信息是否到达了接收者处；另一些时候，发送者需要确认信息已经到达了接收者处，但是不需要等到接受者确认后才能继续下面的工作；还有的时候，发送者必须等到接收者确认收到信息之后才能往下进行工作。

同步模式的操作就是发送者必须接收到接收者的反馈后才能继续往下工作；而不需要接受者反馈信息的工作模式，或者至少不需要接受者立即反馈的，就叫做异步模式。这两

种模式的区别通常决定了某种协议是不是适合某一特定任务。

3．客户端、服务端和对等端

“客户端”、“服务端”和“对等端”等字眼正逐渐和分布计算的某些特定属性联系在一起。实际上，无论是客户端、服务端还是对等端，都只是在通讯中扮演了一个参与者的角色。在每次通讯过程中，这些角色都在不断地变化，这次是客户端的角色，下一次说不定就成了服务端等等。

注意：这些“端”实际上是指正在运行着的线程，而不是狭义地指某种计算机硬件，这些线程有可能存在于同一系统中，甚至在同一进程中。

一个可以称作“服务端”的线程，通常的任务是打开通讯信道，并等待其他线程来与其联系；而主动去联系“服务端”线程来开始进行通信的线程，一般来讲就是“客户端”；至于“对等端”，它既可以充当客户程序，也可以充当服务程序。

4．API——应用编程接口

通讯功能的核心部分通常是由操作系统和网络相关的 API 提供。这两种程序调用大量的通讯函数来完成实际的系统间数据的传输及接收。总的来说，这些低层组件为底层的通讯模块提供了一定层次的抽象，同时也将更高层次的地址标识和数据转换等功能留给高一层的服务模块。直接网络通讯如图 1-1 所示。

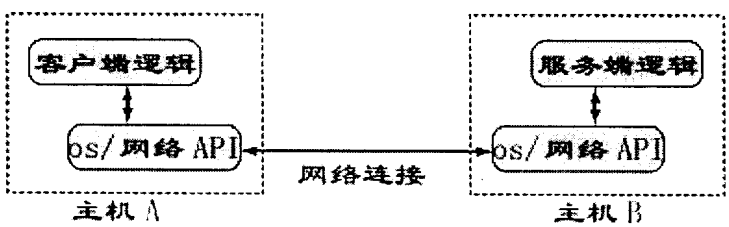

图 1-1　直连网络通讯

5．终端接口

终端，是一种最古老的分布式计算形式，古老到人们甚至认为这根本就不是分布式计算，比如，从哑终端登陆到一个主机系统，或者在工作站上运行的终端模拟器。虽然这些方式不是那么现代，但是效率的确十分高。

一系列通讯协议为这种通讯方式而存在，其中包含 Telnet，rsh 以及 rexec。相对来说，这些协议的原理和执行过程是十分简单的，远程客户端就像直接和主机相连的终端一样，只不过其中包含了一些附加的组件，这些组件允许各个终端通过远程连接的方式和主机进行通讯。每当一个键被按下的时候，客户端就向服务端发送一个标识这个键的数据包。而服务端则按顺序将需要显示的数据反馈给客户端。通常客户终端都是文本界面的，所以有些服务端应用也使用一些颜色和扩展字符来增强客户界面。

这种终端接口的优点在于，在很多情况下，它并不需要应用程序去调用 API 函数，并

且能让程序分布在各处执行而不需要做任何修改。

6．消息

分布式计算方式接下来演化到了消息机制阶段，消息包中包含了消息包的属性标志和具体信息。这样，消息机制就要求服务器上必须要有一层中间处理层来确定消息的路由，以便让它到达正确的接收者处。

消息机制是一种天生的异步机制，因为基于消息的通讯能够很好地和中间层的路由配合。各个消息暂时存放在服务器或路由器上的消息队列中，在此队列中他们等待着一个或几个逻辑上的处理程序对其进行下一步处理。有些处理也许根本就不需要响应某些消息，当然也可以直接反馈给客户端。但是，为了保持逻辑上的抽象吻合，他们仍然需要给服务器发送一个消息，然后通过另一个队列路由返回客户端。

基于消息的结构也可以采用同步模式。一般来讲，在这种模式中，服务器/路由器将消息直接传递给处理程序，然后由处理程序回传处理结果给服务器，再由服务器传给客户端。

还有另外一种混合模式，在这种混合模式中，服务器如前述异步模式进行操作，而客户机则采用同步模式。这种组合让服务端获得了异步操作的高效率，同时也让客户端从同步模式的简单处理机制和安全性上获得了好处。

7．远程过程调用

支持 RPC 的概念十分简单：用以产生进程的调用看起来像是一个普通的调用，而真正的执行是在其他进程中——也许是一个远程系统中的进程。各种 RPC 执行协议都朝着一个共同的目标在发展，那就是用隐藏执行细节来简化进程间通信的复杂性。

RPC 机制的核心概念就是将函数调用产生的数据串行化到一个顺序流中，然后在连接接收端对它进行重组。这两种行为同步发生，就好像传统的过程化编程一样。RPC 客户端进程发出一个看似标准的函数调用，但是，这个调用不会在本地执行，调用参数被打包并传递到一个远程的执行环境当中，在那里它们再被传入真正的执行函数当中。在完成函数执行后，执行结果又被串行化传回客户端，再由客户端函数传给调用者。

当然这种操作也可以用前述的同步方式来实现。

8．客户/服务

如前所述，“客户”和“服务”这两个字眼实质上描述的是在一个通信过程中，参加通信的各方所扮演的不同角色。不过，对于“客户/服务”这个组合起来的词组，从普遍意义上来说，更多地是指一种更高层次上的软件结构，虽然，从概念上来看，“客户/服务”和“客户”、“服务”并无多大区别。“客户/服务”代表着一种处理逻辑结构，在这个结构中，一些较为关键的处理过程在客户端进行，当然客户端也会提交一些操作到服务端去。“客户/服务”方式通常是一种同步模式，因为客户端通常都需要确认提交的操作被服务端执行后，方可继续往下运行。

9．数据库协议

“X/Open 调用级接口标准”（X/Open Call Level Interface）[CLI 96]使用结构化查询语

言为关系数据库管理系统提供了一个标准的接口，而微软的 ODBC 接口则是目前 CLI 标准在实际应用中的最好典范。另外，Sun 的 JDBC 是 CLI 标准在 Java 应用程序中的实际表现形式。

CLI 以及它所支持的程序架构也许在客户/服务计算中应用得最为普遍，它让各种遵循这种标准的应用程序能和大部分数据库进行连接，而不需要了解数据库的具体类型。当然，这种“公用控制器”的缺点也是明显的，那就是它不可能提供访问各种 RDBMS 的专有特性的接口和方法。

依据 CLI 标准开发的 API 表现形式是多样化的，其中既有封闭性不高的消息接口，也有 RPC 远程过程调用接口。类似于消息机制的组件接口所表现出来的是一种混合的同步/异步模式：服务端将初步处理好的不完整结果同步返回给客户端，并且继续独立于客户端异步地完成剩下的工作，在这种模式里，客户端只要得到服务端的初步响应集，就可以继续进行下面的工作，而服务端则将客户端发来的请求放入请求队列中，再一个个地异步处理，处理完了之后再放入反馈队列中；而 RPC 组件接口常常是运用于实时控制，它的操作必须严格同步。

10．中间件

在客户/服务结构中，客户端发起通讯过程，并且直接和服务端进行通讯，而在中间件结构中，客户端和服务端之间还多了一层具有特别功能的“中间件”，其结构如图 1-2 所示。这层中间件可以为通信的双方提供地址和名字的解析，认证和交易语义转换等功能，也可以为其他的和中间件相关的功能提供处理逻辑，例如时间同步、数据格式的转化等。

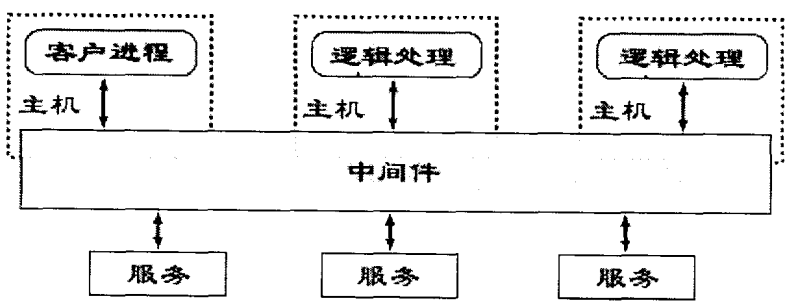

图 1-2　中间件结构

这个额外的层次使多种客户端能够和一个普适的抽象服务器进行交流，而不仅仅是和某一个特定的服务主机或者是某个特定的服务进程进行通信。同时，多种服务也可以将个性特征弱化后，通过这个抽象层提供一个标准的服务接口。这个抽象层让应用程序使用一系列标准的 API 进行开发，而不用再去了解服务端的具体位置和执行细节。对执行细节的封装是中间件的强大功能之一，不过它也有一定的缺陷，那就是客户端无法了解到封装后的服务端究竟会执行怎样的逻辑操作。

11．DCE——分布式计算环境

OSF DCE（注）在[DCE 96]中，将这里描述的很多概念加以整理，并正式制订成了一

组相关的标准。其中以 DCE RPC 标准在业界的应用最为广泛，它表述的是异质执行环境中程序行为一致性标准。另外，DCE 架构也定义了线程、时间、认证、安全，目录服务和命名服务的标准。

因为 DCE 是由主流操作系统厂商的行业协会所支持的，所以这个标准在很多计算平台上都得到了广泛的支持。DCE 核心功能现在已经被几乎所有的 UNIX 系统及其变种所支持，并且，在 PC 操作系统日益普及的今天，DCE 核心服务也在 PC 机上变得越来越普遍。不过，DCE 的这些标准都是以 C 语言中的过程化编程方法为基础制定的，这在一方面也限制它们对于多语言和面向对象的支持。

注意：OSF 是 Open Software Foundation（开放软件基金会）的缩写，现该基金会已改名为 Open Group（开放协会），但 OSF DCE 已经作为商标拥有注册商标权。

12. 可靠消息机制

可靠消息机制是一种消息传递机制，例如 IBM 的 MQSeries 和微软的 MSMQ 等。在可靠消息机制结构最顶部层面，很类似于前面所提到的消息队列结构，但是在这个层次以下则有很大不同。

为了可靠地传递异步消息，人们采用了“存储转发”模式，在这个模式中，需要传递的消息及其附带的地址信息被同步地传入中间层，并在中间层永久存储起来。一但消息进入这种存储状态，中间层就会千方百计地试图将消息发送到它的目的地去，而发送进程此时就可以进行其他的处理。

用这种结构来传递消息的可靠性，在于当前消息持有者总是拥有一份这个消息的永久拷贝，而在消息未转发到下一个目的地并获得接收确认之前，这个拷贝就永远不会被删除。正因为通信链中的每一环都可以保证消息会成功传递到下一站直到终点，所以最初的发送者可以假设：发送出去的消息都会顺利地抵达目的地。那么，基于此假设，发送者就用不着再关心消息的后续情况，而可以直接进行下面的工作了。由于这种结构天生的异步特性，如果发送者需要知道消息到达接收者处的时刻或者具体细节，那么他仍然需要接收者对他反馈确认信息。

13. 分布式对象

分布式对象结构是从中间件概念上发展起来的，它将程序数据封装在具有函数接口的对象之中，其结构如图 1-3 所示。和以前我们设计结构化的程序有些类似——如果程序函数设计得比较好，那么各函数之间是松耦合的，函数执行细节对于外部来说是不可见的；同样，在分布式对象结构中，对象内的执行细节对于调用者来说也是不可见的。并且在分布式对象结构中，对于对象中的方法调用也作了限制，用户不能像调用 API 一样去直接调用这些方法，而只能通过间接的形式进行调用。另外，用户在调用对象的时候也只需要使用对象的引用，而不再需要创建本地实例。

由于在分布式对象结构几乎完全隐藏了对象的执行细节，所以程序执行的地点、平台和编程语言对外界来说都是透明的。不过，这种透明性也带来了一定的耗费，在某些特定的任务中，设计者不得不在更好的性能和透明性之间采取一定的折中。

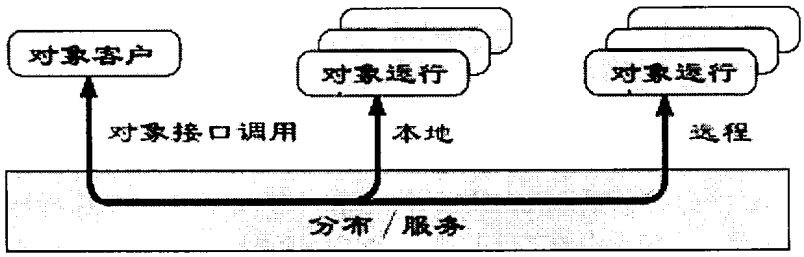

图 1-3　分布式对象结构

14．JavaRMI——远程方法调用

虽然 Sun 的 Java 对于业界来说是相对较新的事物，但是他所支持的平台无关特性、安全性和面向对象特性却迅速在业界获得了广泛的认同。不像其他的编程语言，Java 从一开始就是为一个完整的执行环境所设计的，所以它能够独立于各种底层平台为开发者提供一个一致的抽象界面。

Java 的平台独立性是由 Java 虚拟机（JVM）来实现的，这个虚拟机自身能模拟一种虚拟的计算环境。JVM 虚拟机提供了硬件和各种能运行 Java 的操作系统之间的结合点，因为虚拟机为 Java 程序提供了统一的虚拟环境，所以应用级的 Java 程序可以看作是运行在相同平台上，他们之间的通信就会特别简单。

Java RMI[RMI 97]中制定了一个基于 Java 语言的体系标准，遵循这个标准，人们可以很容易地创建“Java 对 Java”的分布式应用程序。在一个纯粹由 Java 组成的分布式系统中，Java 对象模型在任何时候、任何地点都可以被调用，当然，这不包含 Java 用于多语言环境的情况。不过，Java 所固有的平台无关性仍然允许异质环境的安装和操作。

15．分布式组件对象模型 DCOM

微软对象分布模型的核心协议就是 DCOM，它是微软 COM 集成结构的扩展，它主要是为不同网络环境中的分布对象提供交互的标准。

COM 让客户程序可以动态地连接到对象，然后执行。例如，在客户程序运行的时候，可以将这些对象合并到一个单一的地址空间中。执行的具体过程被包装到动态连接库中（DLL）。由于 COM 为了保证二进制兼容而采用了 C++的虚拟函数表，它实质上是一种程序间的集成方案。这些虚拟函数表（一般都称作“虚表”），用 C 语言中的概念来解释就是函数地址表或者其他类似的说法。COM 接口就是作为指向虚表的指针提供给用户的，如此一来函数的运行细节就变得不可见了。这种特性使二进制形式的 COM 对象可以灵活地进行替换，只要给定的接口不改变，这些二进制代码可以用于实现任何功能。

采用 C++的二进制虚表集成方式，COM 用最小的耗费同时获得了二进制代码的可置换特性，以及内嵌函数调用的超高效率，它相当于 C++语言中的虚函数调用。不过世界上没有免费的午餐，这些改善从另一方面来讲也有着天生的缺陷——因为这里的接口实际上是一个指向单一虚表的指针，那么这些接口就不可能实现多重继承。这就意味着如果要实现

多重继承，一个接口就需要对应多个虚表和多个指针。况且，由于 COM 中不存在任何中间的服务来分离这些函数调用，除了 C++以外的其他语言必须在调用 COM 之前作一些额外的处理。

为了适应逐渐增长的分布于多个主机上的对象的需要（比如，多个物理地址空间），微软开发 COM 的扩展版本：DCOM。作为对 COM 的扩展，DCOM 只是在调用程序和实际的执行接口之间插入了一个转换接口。从这个角度来讲，虽然这种运行机制仍然是以二进制集成方案的形式为基础的，而不是一个抽象的模型，但 DCOM 结构和基于 RPC 机制的抽象模型确实十分类似。

COM+是最近微软宣布的以 COM 为基础的新一代技术。COM+扩展了 COM 的范围，不仅支持多重继承，并且包含新的运行环境，以及语言扩展接口，它让各种语言能更加容易地创建 COM 对象。

16．CORBA——通用对象请求中介结构

CORBA 是对象管理协会（OMG）发布的异质网络分布对象的交互标准。由于这是个与平台无关的对象交互标准，所以得到了业界的广泛认同。

OMG 是一个由业界七百六十多个公司组成的工业协会，这些公司成立 OMG 的目的就是为了共同制定出一个大家都遵循的分布式对象计算标准。OMG（注）的目的则是为了将对象和分布式系统技术集成为一个可相互操作的统一结构，此结构既支持现有的平台也将支持未来的平台集成。而这一切所有努力的结果就是现在大家所见到的对象管理体系（OMA）。CORBA 说明了 OMA 的基础——“对象请求中介”（ORB）标准，在 ORB 标准中，不仅提供了 CORBA 基础架构说明[CORBA 97]，并且还提供了一系列服务，例如安全、交易和消息传递等。

CORBA 使应用程序能够使用一个共同的接口，这个接口可以在多种平台和多个开发工具中用接口定义语言（IDL）来说明。OMG IDL 是语言和平台无关的；而数据及调用格式的转换则是由 ORB 透明地完成。所有的 CORBA 对象接口，以及接口中相关的数据类型，都可以由接口定义语言（IDL）说明。这种对接口的公共定义方法使应用程序能够在不涉及到对象的具体运行方式时对对象进行操作。

从客户端角度来看，一个 CORBA 对象是十分模糊的，所以某个对象的执行地点和执行细节对于使用它的应用程序来讲都是不可见的。一般来讲，一个 CORBA 客户仅仅需要知道怎样通过一些公共的对象接口，例如查询对象接口，来发现或者创建自己需要使用的对象。这种情况下，客户程序很可能对对象的执行地点是一无所知的，而作为客户使用对象的调用依据并不是对象的具体地点，而是各个对象在 CORBA 命名服务[COSS 97]中的对象别名。

所以，客户程序所仅能知道的就是 CORBA 对象的公共接口，通过这个接口，客户程序就可以调用对象的方法和功能。CORBA 还支持运行时的对象接口动态定义、通过它的接口库（IR）进行接口选择调用、以及动态调用接口（DII）。不过，虽然这些特点为程序在运行时访问 CORBA 对象提供了几乎是完全的动态配置能力，但是在实用中由于语义上的障碍，这些特性实际上仅在少数场合才有其用武之地。

注意：微软也参加了 OMG，但并未提交任何技术给 OMG。相反，微软大力推广的是 Windows 为中心的 DCOM 标准。

上述的很多理论和概念在实际的分布式结构中很多都是在混合地运用，并且其中的一些结构在应用中是互为基础的。

某些技术，虽然很陈旧，但是仍然很有必要使用。所以，没有哪个大型机或者 UNIX 系统的系统管理员会放弃文本终端来进行拨号远程管理的方式。类似地，频繁地执行大吞吐量或者低层次的系统通讯操作，也是一些低层次操作系统和网络接口最能胜任的任务。

不过，在费用和便捷成为使用中的主要因素时，一般来说标准化的体系结构则是一个较好的选择。在这种情况下，人们通常将标准化的体系分成两种情况来考虑：异步的和同步的。根据异步和同步的进一步细分。我们可以得到程序中需要使用的相应的过程化的方法和面向对象的方法。

对于异步通信来说，基于消息的体系常常是最为合适的。在某些情况下，由于整个系统的可靠程度本身就比较高，所以消息传送的可靠性要求不是那么高；但在另外一种情况下，消息传送的可靠程度要求很高，那么，可靠的存储转发中间件就是解决这个问题的答案。

而在同步模式中，我们常常使用过程化的编程，DCE RPC 是通常情况下的最好选择。它证明，实际中大量而广泛运用的具有 C 语言接口的体系在 C 或者 C++应用程序中十分易于使用。而在其他的除 C 语言外的系统中，DCE RPC 的支持是不那么完整的，此时同步消息机制就是我们所需要的答案。

最后也是最重要一点，也就是面向对象的分布式体系。毫无疑问，在前面所阐述的三种结构中，语言和平台无关的 CORBA 提供了最大的灵活性和通用性。当然，这种无关性也产生了资源耗费。不过，一个不那么灵活的结构，如果它不支持未来的一些要求和特性，从长远来看，会有更多的麻烦。

微软的 COM/DCOM 方案能够获得成功的最主要因素之一，就是几乎所有运行着 Windows 的个人电脑都或多或少地使用了内建的 COM 支持，并且大多数 32 位版本的 Windows 还支持 DCOM，几乎所有的 Windows 开发工具提供相当容易集成 COM 组件的封装包，这使得在 Windows 环境下 COM 为基础的技术在分布标准中成为一个强有力的竞争者。

使用 Java RMI 的原因非常简单，因为 Java RMI 很容易使用。因为它仅仅支持 Java 对象，所以它能在很少影响开发资源的情况下，很容易地采用 Java 模型。不过，因为 CORBA 支持是建立在 Java 环境中的，所以 RMI 由于没有广泛的使用基础而失去了一定的优势。但是，在纯 Java 系统中，基于 RMI 系统的易用性是无可怀疑的。

上述各种结构都有大量的支持者，所以无所谓哪种结构更好或者哪种结构更差，并且虽然这里提到了一些广泛使用的分布式计算结构，但是其实还有更多的结构尚未在这里阐述。另外，特别值得一提的是，可靠消息传递机制是一个正在快速发展的领域。

关于 COM 和 CORBA 的更加详细的比较可以参考[Quoin 98]。当然，上述的各种结构的比较都可以在互联网上找到，这里就不赘述了。