CORBA 原理及应用

朱其亮
郑　斌　编著

北京邮电大学出版社
·北　京·

内 容 简 介

本书系统介绍了 CORBA 的组成、工作原理、静态和动态调用技术以及可移植对象适配器。详细介绍了接口定义语义及其向 C^{++} 和 Java 语言的映射,以及在静态调用方式下 CORBA 的编程方法。简要阐述了 CORBA 的通信模型、通用 ORB 互通协议及其派生的特定互通协议。此外,深入讨论了 CORBA 常用服务,如命名、事件、通知、对象交易和事务处理的原理和应用流程。最后,探讨了 CORBA 技术在电信领域,如网络管理和智能网中的应用。

本书可作为高等院校计算机专业、通信专业和信息技术专业的大学生和研究生的教材和教学参考书,也可作为相关领域的研究人员、软件开发人员和工程技术人员的参考书。

图书在版编目(CIP)数据

CORBA 原理及应用/ 朱其亮,郑斌编著 . —北京:北京邮电大学出版社,2001.9
ISBN 7-5635-0444-3

Ⅰ.C... Ⅱ.①朱 ...②郑 ... Ⅲ. 分布式处理系统—应用软件,CORBA Ⅳ.TP316.4

中国版本图书馆 CIP 数据核字(2001)第 051968 号

书　　名: CORBA 原理及应用
作　　者: 朱其亮　郑　斌
责 任 编 辑: 周　明
出　版　者: 北京邮电大学出版社(北京市海淀区西土城路 10 号)
　　邮编: 100876　电话: 62282185　62283578
　　网址: http:// www.buptpress.com
经　　销: 各地新华书店
印　　刷: 北京源海印刷厂
印　　数: 1—3 000 册
开　　本: 787 mm×1 092 mm　1/ 16　印张 25.5　字数 604 千字
版　　次: 2001 年 10 月第一版　2001 年 10 月第一次印刷
书　　号: ISBN 7-5635-0444-3/ TN·209
定　　价: 46.00 元

序

最近几年,在计算机软件领域,分布式处理技术与面向对象技术的融合成为引人瞩目的潮流。CORBA(通用对象请求代理体系结构)在开发大型分布式软件的过程中起着越来越重要的作用。

CORBA 有着良好的开放性和扩展性。近年来,CORBA 又将其他中间件的特色吸收到所定义的服务之中,同时它也注重不断融合最新出现的技术,如 Java,UML,XML,组件技术和移动代理(mobile agent)等。

CORBA 技术是先有标准,后有产品,这同许多与 CORBA 竞争的其他中间件有着很大的区别。一个理想化的标准在成为产品的过程中,难免会出现一些性能不够理想且难以互联互通等问题。随着时间的推移,这些问题终将得到解决,其理论上的优势会更加明显地突现出来。

事实已经证明,在大型分布式异构软件系统的领域(如电信和金融系统)中,CORBA 正在得到越来越广泛的重视和应用。尤其在通信软件领域,无论是电信管理网还是智能网,有关 CORBA 技术的研究和工程开发呈现出蓬勃发展的趋势。CORBA 正以其年轻的活力影响着 21 世纪的软件技术格局。

本书的作者在规范研究、教学实践和项目开发的基础上,编写了这本系统介绍CORBA原理和应用的著作,这一工作对推广 CORBA 技术和提高我国通信软件(特别是大型分布式软件系统)的开发水平必将起到积极的推动作用。本书对于高等院校的计算机专业、通信专业和信息技术专业的大学生、研究生以及相关领域的研究人员、软件开发人员和工程技术人员来说具有很大的参考价值。

祝贺本书的出版。

中国科学院及中国工程院院士
北京邮电大学教授　　陈俊亮

2001 年 8 月

前　言

利用前言这一小小的篇幅,说明以下 3 个问题:CORBA 是什么,为什么要写这本书,以及这本书写了什么。

1 . CORBA 是什么?

打开这本书,对于没有接触过 CORBA 的读者来说,很容易将 CORBA 误解为一种编程语言。然而,它不是,虽然 CORBA 也提供了描述对象接口的接口描述语言,并且通过编译器自动生成码根和框架代码,为最终编写客户应用程序和服务器应用程序提供了编程基础。

从本质上讲,CORBA 是一种结构,一种通用的对象请求代理体系结构。它把各种对象的操作和属性封装在不同的接口之中,通过对象请求代理来调用接口中的操作,完成指定的功能。这种体系结构为开发大型的分布式异构应用软件系统提供了独立于硬件平台、独立于编程语言、独立于计算机操作系统和独立于网络传送协议的平台。另一方面,CORBA 还提供了面向对象的分布式软件所需要的许多通用服务,如命名服务、事件服务、通知服务、交易服务、对象事务处理服务、生命周期服务以及消息服务等等。从这个意义上讲,CORBA 不仅是一种体系结构,同时又是一种服务环境。

2 . 为什么要写这本书?

CORBA 以其独特的优势,在开发大型分布式软件系统的过程中,发挥着日益重要的作用。但是,正如人们所普遍感到的那样,CORBA 比较抽象,不容易理解和掌握,这是其一。目前,国内出版的有限的几种 CORBA 方面的书籍,大部分是从国外翻译过来的,在数量上和取材上还远远不能满足在中国推广 CORBA 技术的需要,这是其二。事实上,近年来 CORBA 技术的应用已经在许多领域悄然兴起,并正在得到越来越广泛的应用,而 CORBA 技术本身也在应用中逐渐地得到完善。因此,客观上,需要中国的软件工作者自己编写更多的有关 CORBA 原理和应用方面的书籍,以适应 21 世纪通信和信息业发展的需要,这是其三。

基于以上的原因,我们在教学实践和项目开发的基础上,编著了这本系统介绍 CORBA 技术的书籍,希望能起到抛砖引玉的作用,吸引更多的软件人员来关注和发展 CORBA 技术,从而进一步提高开发通信软件,特别是大型分布式软件系统的水平。

3.这本书写了什么?

《CORBA 原理及应用》一书包括了 4 个部分:

第 1 部分是 CORBA 的基本原理。包括 CORBA 的基本概念、基本组成、工作方式、接口定义语言及编程方法。

第 2 部分是 CORBA 更深入的内容。包括动态调用、接口库、可移植对象适配器以及 CORBA 的通信模型。

第 3 部分是 CORBA 提供的最常用的服务。包括命名服务、事件服务、通知服务、对象交易服务和对象事务处理服务。

第 4 部分是 CORBA 技术在电信领域中的应用,包括联合域间管理和智能网互操作。

本书最后的附录,列入了 IDL 向 Java 语言的映射以及最小 CORBA 等内容。

当然,以上 4 个部分的内容不可能涵盖 CORBA 的所有方面,但它是 CORBA 最核心的内容。在叙述方法上,作者力图从基本概念入手,先建立总体印象,然后,对复杂问题进行深入浅出的解释,对难懂的和抽象的概念进行具体化的描述,从而改变直接阅读技术规范时难以理解的局面;同时,在本书的编写过程中强调了实用性,即原理与应用相结合的原则。例如在介绍静态调用时,配合实例具体说明了静态环境下编程的方法和步骤;在讨论了可移植对象适配器的原理后演示了可运行的实例,这些都将帮助读者把原理与实际应用结合起来。基于同样的考虑,本书的最后两章给出了 CORBA 应用的例子,读者从中可以得到基于 CORBA 体系结构实际应用的启示,以便举一反三,开发出自己的应用。此外,本书所用的参考文献包括了最近颁布的 CORBA 标准。目的是尽量将最先进的内容介绍给读者,以期拓宽读者知识面和加强对 CORBA 理解的深度。

总之,本书既可作为高等院校的教材,又可作为软件开发人员和工程技术人员的参考书。

在结束前言的时候,作者感谢通信和信息领域的同行屠翔、王栋和林起泰对编著本书给予的鼓励和支持,对研究生黄秦、李晓利、丁华和刘晓菲等利用业余时间参与本书的录入和制图工作表示衷心的感谢。最后,对作者的家人们在本书成书过程中给予的支持和帮助表示由衷的谢意。

我们热切地期待着来自同行们的批评和指正!

作　者

2001 年 8 月于北京

目　录

1 分布式系统与中间件

本章首先介绍集中控制系统到分布式系统的演进历程。然后,讨论中间件的概念和分类,这里对分布式计算环境、面向消息的中间件、事务处理中间件以及面向对象的中间件作了简要的描述,由此说明了分布式处理技术与面向对象技术的融合是软件发展的必然趋势。

1.1 分布式系统的演进

1.1.1 集中控制式系统

早期,在面向过程的语言环境中,本地进程内调用是提高效率的最常用的一种方式,它不需要跨越进程,更不用跨机器平台,采用单一语言即可实现。这种形式的调用实现简单、方便和高效。属于这一种调用形式的应用大量存在。我们最早学习计算机编程所接触到的语言,如汇编、PASCAL 和 C 等,常使用这种调用形式。

到了 20 世纪 80 年代,面向对象技术逐渐兴起,面向对象技术以其封装、继承和多态的特性顺应了软件发展的新的需求,良好的重用性使得软件开发在理论上像搭积木那样简单。适用于这一技术的语言现在有很多,最有代表性的是 C^{++} 和 Java。

在集中控制式系统中,不论是面向过程的,或是面向对象的调用都发生在本地进程内,如图 1.1 所示。图中较大的矩形表示主机范围,较小矩形表示进程范围,圆圈表示对象。

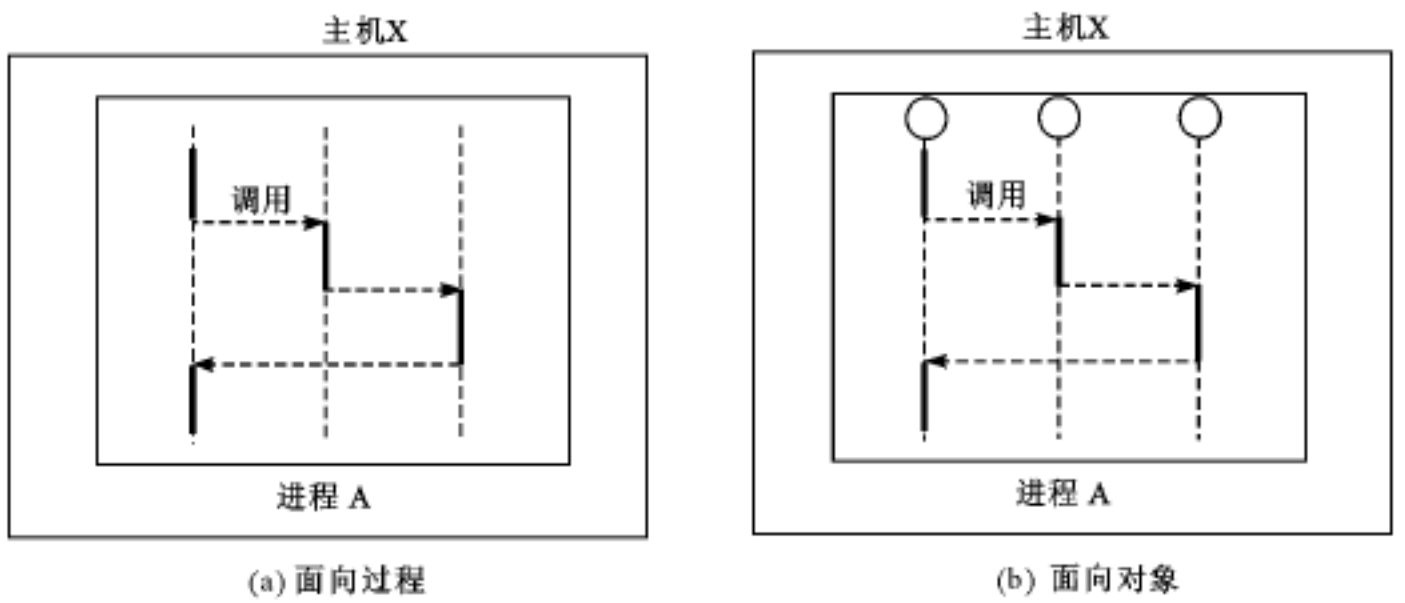

图 1.1 本地进程内调用

随着软件复杂度的提高,某些复杂工作不能由单一进程来完成,而是需要多个进程彼此分工、协同工作。同时,由于多个进程可以利用并行机制提高程序效率,所以不仅对面向过程的语言,而且对面向对象的语言都提出了进程间通信的需求。本地进程间通信(见图 1.2)不仅可以在两个进程内传递必要的信息,还可以用来协调两个进程间的步调,在分时操作系统中合理分配 CPU 时间,是并行处理一个不可缺少的机制。

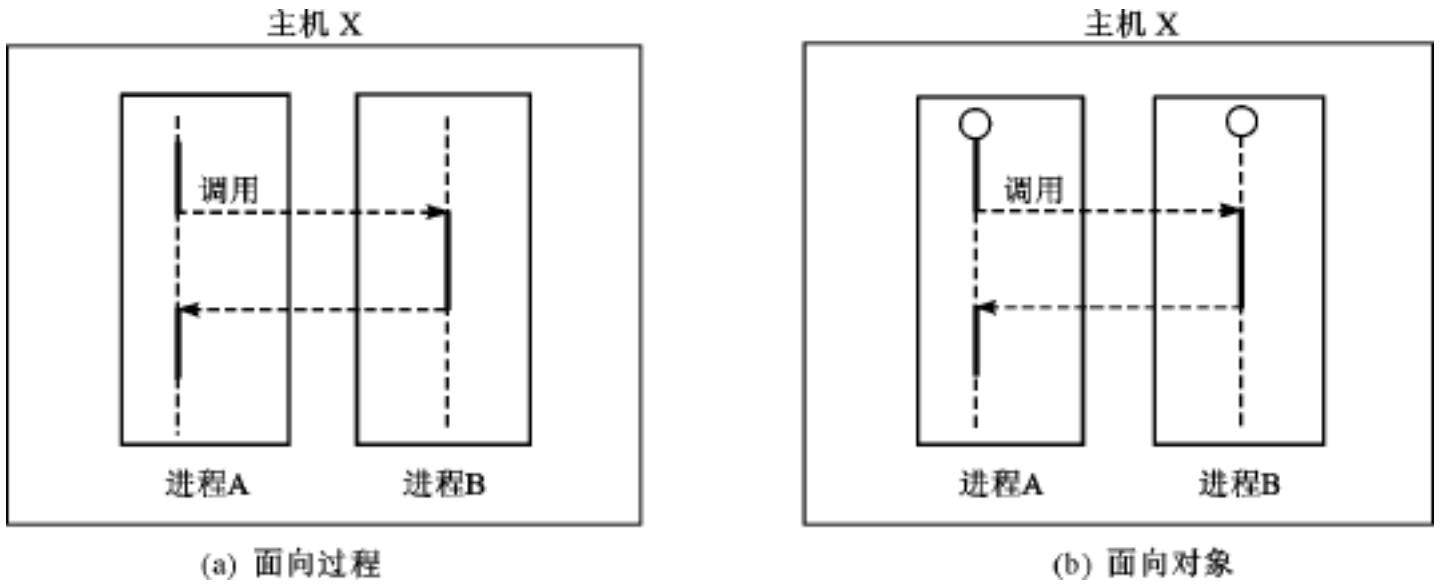

图 1.2　本地进程间调用

进程间通信又称 IPC(Inter Process Communication),在 UNIX 环境下,常用的进程间通信的手段有:

信号(signal)
管道(pipe)
共享内存(shared memory)
信号量

当某个事件发生时,该事件以信号的方式发送给某个进程,进程得到信号后,把它交给特定的信号处理器进行处理,或者阻塞,甚至丢弃;管道用于连接两个进程的输入和输出,使进程之间交换信息就像是读写文件那样简单;共享内存是在内存中设置一个公用的区域,供通信的进程共同访问,达到互通信息的目的;此外还有信号量,通过对信号量的 P,V 操作,各进程可以对临界区的访问进行同步控制。实现这些机制的 C 语言标准函数在 POSIX 标准和 UNIX 系统 V 中都作了明确的定义,这些函数同时也可用于 C^{++} 编程之中。但是无论使用信号量还是共享内存或其他方式,通信双方必须位于同一机器平台之上,并且使用相同的函数接口。

UNIX 之外的其他操作系统,或多或少地都提供有本地进程间通信的手段。

1.1.2　分布式系统

最早的分布式系统可以追溯到上个世纪六七十年代的主架(mainframe)型计算机。那时的计算机系统是由一个巨无霸型的小型机(例如 VAX-11)、中型机或大型机加上与其相连的多个外设终端所组成。终端(或称为哑终端)只负责与用户的输入输出交互,而实际的逻辑处理和计算任务都会传递给主架机处理,最后得到的结果由终端通过屏幕或其他

方式返回给用户。主架机虽然昂贵,但是它可以同时支持很多个终端用户,而终端的价格就相对便宜得多,这种体系结构在当时计算机还远没有成为个人消费品的情况下,是一种非常不错的选择。主架型计算机虽然采取了分布的形式(主架机/ 终端),但严格地讲,它是集中控制向分布控制的过渡。

到了 80 年代,情况发生了变化,PC 机(这里泛指个人电脑,而不仅限于 WINTEL 架构的 PC 机)开始普及,价格越来越便宜,性能也越来越好,早期很多只能由主架机完成的工作可以由 PC 机轻松地完成,这仿佛是很多任务和处理从集中的主架机“流”到了终端那里。事实上,非智能化的终端已被智能化的 PC 所取代,而主架机则让位给了服务器。这种由前端 PC 机和后台服务器组成的体系就是客户机/ 服务器(client/ server)的体系结构,由于服务器比主架机要便宜许多,能被更多的企业所接受。

同一时期,计算机网络的发展也很迅速。早期的主架型机器的星型拓扑结构演变为丰富的网络拓扑结构,客户机和服务器成为网络上对等的节点。客户机/ 服务器不仅在形式上是分布的,在处理上也确实是分布的,客户机完成大多数界面交互和一部分逻辑处理,而将较为复杂和专业的处理交给后台的服务器。服务器的性能一般较客户机要高,不同的服务器往往负责不同的专业事务,如数据库服务器、Web 服务器、文件服务器等等。客户/ 服务器体系结构催生了大量的分布式实用技术和协议,如 ODBC, RPC, HTTP, FTP, TELNET 等等。

进入 90 年代以后,客户/ 服务器体系结构在两个方向上发生了变化。

第一,两层(two-tier)结构向多层(multi-tier)结构的变化。人们发现,将太多的逻辑交给客户端去处理会降低应用的扩展性,因为,每次对应用的逻辑进行的改造都不可避免地要修改客户端的程序,给程序的升级和分发带来麻烦。那么,是否可以将事务逻辑从客户端剥离出来,专门由一个中间层来处理,而将传统的数据库服务或文件服务放到最后一层去做呢?答案是肯定的。为此,需要给客户机“减肥”,而 Web 浏览器的风行终于使人们看到了希望。这种“瘦客”户的模式使我们想起几十年前的主架型计算机中的哑终端,历史绕了一个弯,又螺旋上升到原先的位置。

第二,客户机和服务器之间的界限日渐模糊。从性能上说,早期的主架型机与哑终端有天壤之别。与此类似,早期的客户机与服务器之间也有很大的差异,而到了现在,随着应用的复杂化,一台机器可能在一个时候是客户,在另一个时候是服务器,客户机和服务器与机器的物理实体之间的对应关系越来越不固定,正因为如此,我们现在往往把一段程序(或进程)所扮演的角色称为客户或服务器(即使这样,也有问题,因为一段程序往往一会儿作客户,一会儿又作服务器),而不再流行把某一台物理机器称为客户机或服务器。例如,在一个 3 层的 Web 应用中,位于中间层的 Web 服务器可能需要存取第 3 层数据库里的资料,这时,它就是第 3 层的客户,第 3 层是数据库服务器,同时,作为 Web 客户的第 1 层的机器,此刻可能正运行着 NFS 文件服务后台进程,充当着第 2 层和第 3 层的文件服务器。客户/ 服务器体系结构的进化很像人类社会自身的进化,社会分工越来越细,每个人在专业领域为其他的人提供服务,同时在其他领域依赖着别人的服务。

从以上的讨论可以看到,没有远端调用,就不可能有分布式系统的发展。位于两台或多台机器之上的进程之间的调用可以是面向过程的,也可以是面向对象的,如图 1.3 所示。

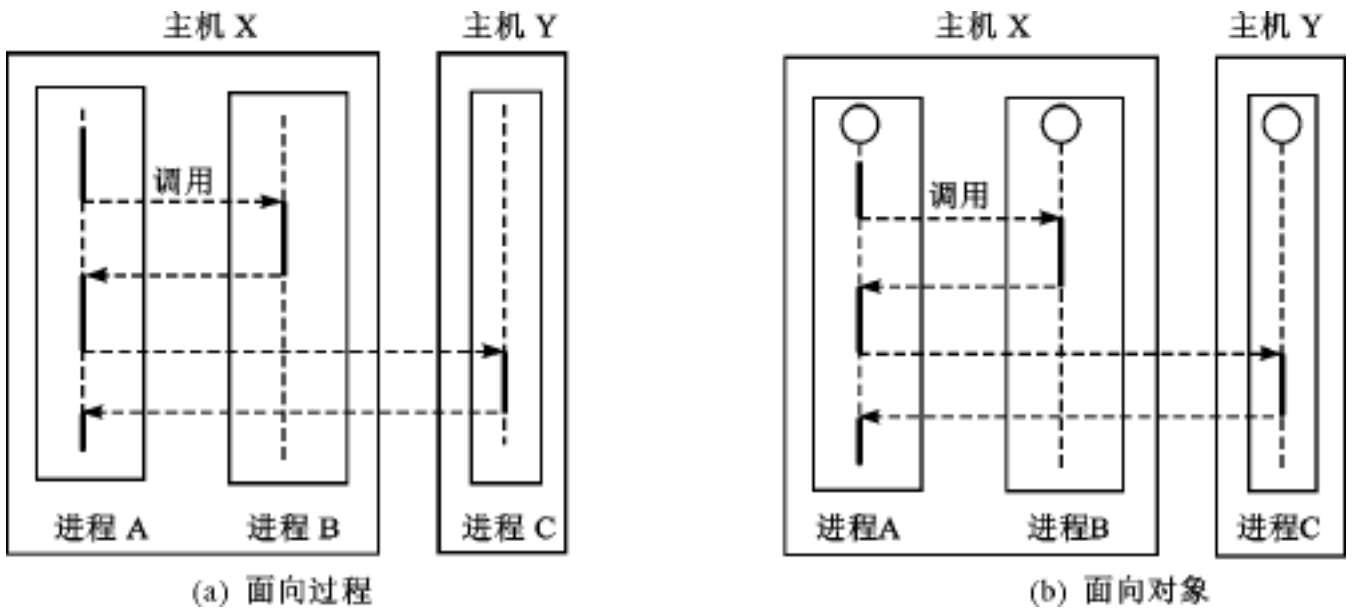

图 1.3　分布式调用

1.2　中间件

1.2.1　中间件的概念

中间件(middleware)一词每天都被不同的人群所提及,中间件的产品也大量出现在软件市场,但中间件的定义却一直比较模糊,导致中间件的分类也不很统一。也许正因为如此,才使人们不断地设计和开发新的产品,并冠以中间件的标签,推动着中间件的内涵和市场不断地向前发展。

我们试图给出中间件的一个定义。所谓中间件,就是位于操作系统和应用软件之间的一个软件层,它向各种应用软件提供服务,使不同的应用进程能在屏蔽掉平台差异的情况下,通过网络互相通信。通常,在实际使用中,把一组中间件集成在一起构成一个平台(包括开发平台和运行平台),其中必须要有一个通信中间件完成中间件之间的通信。从这个意义上讲,中间件应包括平台和通信两个部分。图 1.4 是对这一定义的解释。

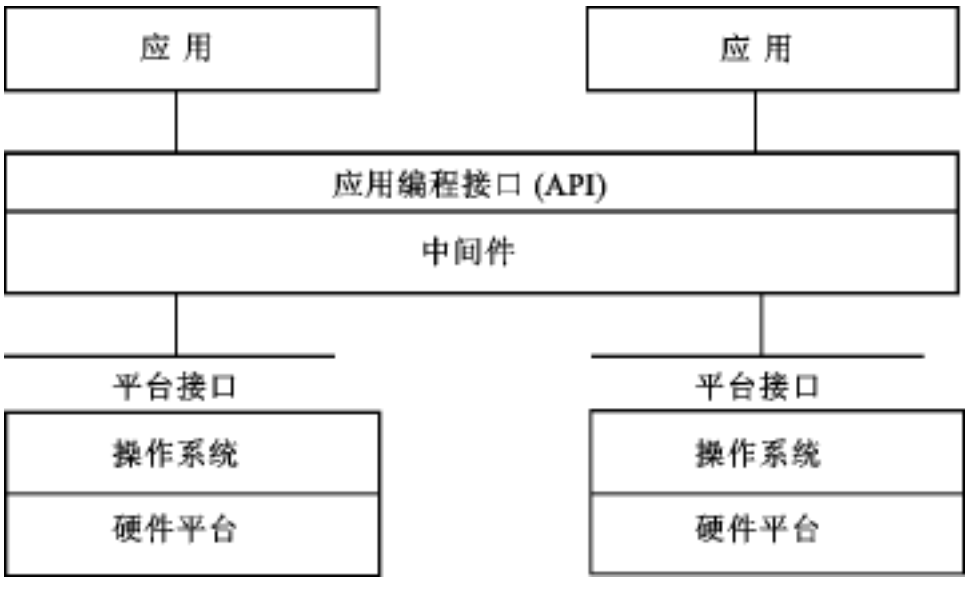

图 1.4　中间件示意

中间件的思想其实并不复杂。假设我们有 n 个应用，m 个操作系统，为了使所有的应用在所有的操作系统上都能工作，就可能需要 $n \times m$ 个接口。而且，每引入一个新的操作系统，就要重新改写 n 个应用的源代码；每开发一个新的应用，就要考虑实现 m 个不同的版本，以工作于 m 个操作系统之上。

中间件使这一切变得简单。开发应用程序时不必再关心底层操作系统的类型，而只需专心于应用的逻辑处理(当然，这只是一种理想状况)。

中间件的引入使原来的网状接口类型变成了沙漏状接口类型，如图 1.5 所示。接口数目从 $n \times m$ 降到了 $n + m$。当 n 和 m 都很小的情况下，这种差异并不明显，但随着 n 和 m 不断地增大(正如现实世界正在发生的那样)，这种差异将极大地增加开发应用软件的困难，并降低系统整体运行的效率和性能。因此，中间件的出现是分布式系统发展的产物，是软件构架演进的必然。

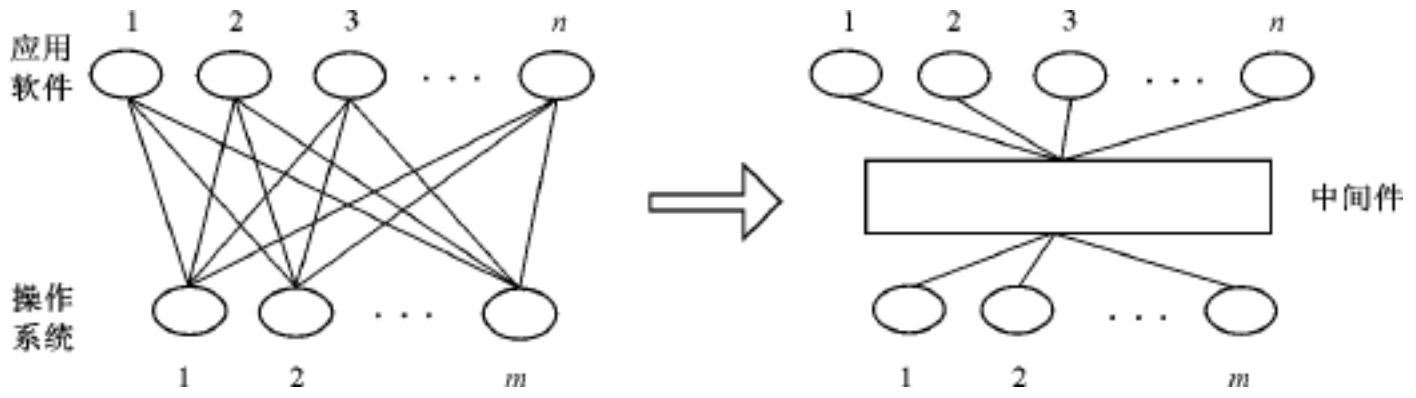

图 1.5　中间件的加入减少了接口数目

1.2.2　中间件的分类

给中间件分类是一件困难的事情。因为如果从功能上划分，很多中间件之间有重复之处，而且随着时间的发展，中间件标准也在不断地完善，实现了原来由其他中间件提供的功能。因此，对中间件的划分不能采取一刀切的方法，而只能根据其“侧重点”进行分类。

国外有人建议将中间件分为 6 类：异步远端过程调用(asynchronous RPC)，发布/订阅(Publish/ Subscribe)，面向消息的中间件(MOM)，对象请求代理(ORB)，面向 SQL 的数据访问中间件(SQL-oriented Data Access Middleware)和同步远端过程调用(synchronous RPC)。其分类依据是中间件的可扩展性(scalability)和可恢复性(recoverability)。也有人把中间件分为 4 类：TP(Transaction Processing)，RPC，MOM 和 ORB。

下面对分类作一简要说明：

异步 RPC 允许客户在向服务器提交申请之后，无需等待操作的完成，就可以转而做其他的事情。现有的产品有 NobleNet RPC。

发布/订阅中间件中的发布方在某类事件出现的时候，向所有事先订阅这一类事件的订阅者发布通知，订阅者在收到通知以后可以决定是否对通知做出回应。这种通信方式也是异步的，因为事件的订阅方在订阅了事件以后，可以着手去做别的事情，而发布方在发布通知以后，也不必等到订阅方给出回应信息。发布/订阅

中间件的产品有 TIBCO 的 TIB/ Rendezvous 和 Active 的 ActiveWeb。

MOM 中间件同发布/ 订阅中间件有些类似,同样属于异步通信,消息的发送方将消息发送到指定的消息队列中,消息在消息队列中保存,直到消息的接受方取出这些消息。MOM 的特点是它能够在系统出错以后提供对消息队列的恢复,从而保证数据的完整性和可靠性。主要的 MOM 产品有 IBM 的 MQSeries, Microsoft 的 MSMQ, Talarian 的 SmartSockets,东方通科技的 TongLINK/ Q 和 BEA 的 MessageQ。

ORB 是对象请求代理的简写。它在面向对象的系统中提供了跨硬件平台、跨操作系统、跨编程语言和跨网络协议的远端调用功能,实现了应用的即插即用。

可见,对中间间的划分没有统一的标准可以遵循,因为中间件并不是在有了一个先验的理论和标准后发展的,因此对它的划分也只是对既成事实的罗列和归并。

目前,最为有名的中间件有 OSF(Open Software Foundation)的 DCE(Distributed Computing Environment), OMG(Object Management Group)的 CORBA, Microsoft 的 COM/ DCOM(Common Object Model / Distributed Common Object Model), MOM(Message-Oriented Middleware), RPC(Remote Procedure Call), TP Monitor(Transaction Processing Monitor)等等,下面分别作一简介。

1. 分布式计算环境(DCE)

DCE 由 OSF 颁布,具体由 Cory Vondrak, TRW, Redondo Beach, CA 等公司负责制订。其主要目的是在异构的网络环境下提供互操作能力。

在 1992 年,OSF 首次颁布了对应 DCE1.0 标准的源代码,这些源代码是 DCE1.0 标准的参考实现,各个 DCE 的实现厂商可以根据自己的需要对其进行更改,这一过程大概经历了 12 个月,最终在 1993 年夏天,DCE1.0 标准才得到各个厂商的实现支持。即使如此,这些实现基本上基于 UNIX 平台,而且也没有覆盖 DCE 的全部功能。到了 1994 年,支持 DCE 的厂家达到 14 个,而这时 OSF 也颁布了 DCE1.1 版。随着面向对象的 ORB 的日渐盛行,DCE 的一些缺憾也暴露出来,例如它的参考源代码是基于 C 语言的,无法提供对面向对象语义的支持。为此,OSF 在 1996 年 3 月颁布了 DCE1.2 版,在这一版本中,除了在可靠性和性能上有所增强以外,其最大的改进就是增加了对面向对象语义的支持,支持 C^{++} 的 IDL 使得类之间的继承和对象引用这些概念得以体现。

DCE 的体系结构示于图 1.6。

DCE 的服务分为两大类:基础分布式服务(Fundamental Distributed Service)和数据共享服务(Data-sharing Service)。

(1) 基础分布式服务

远端过程调用(RPC)。它提供了 DCE 的分布式通信机制。前面已经指出,在很多对中间件的分类中,RPC 本身也可作为一种中间件。此外,RPC 不仅可用于构筑 DCE,在后面将看到,它也可用来作为 CORBA 的底层通信平台。

分布式时间服务(Distributed Time Service)。在分布式环境中保持各节点之间的时钟同步对实现分布式应用是至关重要的,这一服务由分布式时间服务提供。

局部目录服务/ 全局目录服务(Cell Directory Service / Global Directory Service)。在许多分布式环境中(操作系统、中间件、OSI 等)都提供目录服务,它就像现实世界中的电话查号台。在 DCE 中,目录服务被分为局部目录服务和全局目录服务,它