分布计算环境

王 柏　王红熳　邹 华　编 著

北京邮电大学出版社

分布计算环境

王 柏 王红熳 邹 华 编著

北京邮电大学出版社
·北 京·

内 容 简 介

随着计算机应用范围的扩大,许多应用程序需在网络环境的异构平台上运行。分布计算环境基于面向对象技术及 Client/Server 结构的分布式计算技术,提供了网络环境下不同软、硬件平台资源共享和互操作的有效手段。

本教材全面系统地介绍了分布计算技术的基本概念、模型、代表性技术、体系结构框架以及最新的发展趋势。内容包括 Client/Server 结构,中间件技术,开放分布处理模型(ODP),公共对象请求代理结构(CORBA),智能代理技术(Agent),下一代的软件开发模式——软件构件结构,分布计算技术在电信领域的应用,电信信息网络体系结构(TINA),主动网络技术等内容。

本教材适用于信息与电子类专业,特别是计算机和通信专业的高年级学生及研究生,也可供相关软件开发技术人员和管理人员参考。

图书在版编目(CIP)数据

分布计算环境/王柏,王红熳,邹华编著.—北京:北京邮电大学出版社,2000.8
ISBN 7-5635-0438-9

Ⅰ.分... Ⅱ.①王...②王 ③邹... Ⅲ.分布-计算技术-高等学校-教材 Ⅳ.TP30

中国版本图书馆 CIP 数据核字(2000)第 45349 号

书　　名	分布计算环境
编　　著	王　柏　王红熳　邹　华
责任编辑	张学静
出版发行	北京邮电大学出版社
社　　址	北京市海淀区西土城路 10 号　邮编:100876
电　　话	(010)62282185
网　　址	www.buptpress.com
电子邮箱	publish@bupt.edu.cn
经　　销	各地新华书店
印　　刷	北京市忠信诚印刷厂
开　　本	787×1 092 mm　1/16
印　　张	11
字　　数	280 千字
版　　次	2000 年 8 月第 1 版　2000 年 8 月第 1 次印刷
印　　数	1—3 000 册
书　　号	ISBN 7-5635-0438-9/TN·195
定　　价	22.00 元

前　言

随着网络技术的发展，分布处理技术也越来越受到重视，因为人们使用网络的目的是要充分利用网络系统可支持的各种服务，达到预期的目标。但是，在网络分布系统中所需要考虑和解决的问题要远远多于和复杂于网络系统中单机所面临的问题。例如需要解决在网络分布系统中各个层次上的异构问题，包括向下层次上的异构硬件平台、异构的操作系统、异构的网络通信系统，向上层次上的不同应用系统之间的集成问题。在大量用户努力把自己连接到因特网、广域网的今天，解决这一问题的重要性日益突出，如果解决的不好，将成为深化网络应用的障碍。因此，分布处理技术已成为计算机发展研究的热点。

目前，分布处理技术正在向开放式分布计算发展。分布计算环境基于面向对象技术及 Client/Server 结构的分布式计算技术，提供了网络环境下不同软、硬件平台资源共享和互操作的有效手段。为使读者了解开放式分布计算环境这一计算机领域新技术及其发展趋势，我们结合几年来的教学、科研实践编写了本教材。全书共分九章，涵盖了分布对象计算的概念（如 Client/Server，中间件），开放式分布计算模型（如 ODP）及代表性技术（如 CORBA，Agent），分布计算技术在 Internet 中的应用（如对象 Web），分布计算技术在电信领域的应用（如智能网、电信管理网），以及未来基于分布处理技术的统一的电信信息网络体系结构（TINA）。同时，为开拓学生视野，本教材还介绍了主动网络技术，以使学生了解分布计算领域的最新进展。

本教材由王柏、王红熳、邹华编写。书中的许多内容取材于北京邮电大学国家重点实验室“通用个人通信”项目组师生历年来的研究成果，编者在此特别感谢苏森博士、朱于军博士、陈君博士、刘波同学等，他们在 CORBA 技术、智能代理技术、智能网技术、主动网络技术等方面进行的研究工作为编写本书提供了极大的帮助。同时，北京邮电大学出版社为本教材的顺利出版付出了很大努力，谨此一并致以诚挚的谢意。

由于编者水平所限，书中的不足之处在所难免，望广大专家和读者给予批评指正。

编　者

2000 年 8 月

目　录

第1章 基础知识

1.1 分布计算技术的发展历程

分布计算(Distributed Computing)技术是近20年来影响计算技术发展的最活跃因素之一,它的发展经历了两种不同的技术路线,一种是理想的技术路线,一种是现实的技术路线。

理想的技术路线试图在互连的计算机硬件上部署全新的分布式操作系统,全面管理系统中各自独立的计算机,呈现给用户单一的系统视图。在20世纪80年代,学术界普遍追求这一目标,尽管产生了许多技术成果和实验系统,但却没有被用户和市场接受。面对现实情况,人们开始探讨新的解决方案。

现实的技术路线是在网络计算平台上部署分布计算环境(也称为中间件),提供开发工具和公共服务,支持分布式应用,实现资源共享和协同工作。在20世纪90年代,工业界普遍遵循这一技术路线,产生了一系列行之有效的技术和广为用户接受的产品。

当前,人们所说的分布计算技术通常是指在网络计算平台上开发、部署、管理和维护以资源共享和协同工作为主要应用目标的分布式应用系统。

分布式计算的特点决定了分布式应用程序的组成部分将分布在异构网络环境中,因而对分布式应用程序自身的可扩展性、高可用性、管理的方便性、高性能和数据完整性等都提出了新的要求。

1. 分布计算技术的发展历程

从80年代中期开始至今,分布计算技术已经走过了第一代,目前正处于第二代的成熟期,并且开始孕育第三代(参见表1-1)。

表1-1 分布计算技术发展的三个阶段

	第一代 (80年代中至90年代初)	第二代 (90年代)	第三代 (2000年以后)
面向的主要问题	信息共享	异构环境下的应用互操作	智能化的协同工作
体系结构	经典的客户/服务器计算模型	面向对象的多层客户/服务器模型	自主的多Agent模型
关键技术特点	运用传统的计算概念和设施(如过程调用和文件)	将面向对象技术应用于分布计算	面向Agent的拟人化的交互环境
成果	能够提供丰富的分布式系统服务、良好的分布系统管理和典型的分布系统应用	已经成为建立集成构架和软构件标准的核心技术	概念验证系统令人鼓舞,尚未达到广泛应用于协同工作的成熟程度

(1) 第一代

80年代中后期,以支持信息共享的应用需求为核心,形成了面向过程的第一代分布计算技术。在第一代分布计算技术的推动下,90年代初出现了从集中计算模式向分布式客户/服务器计算模式转移的热潮。在分布式客户/服务器计算机系统的建立及其应用系统的开发过程中,人们逐渐体会到分布式系统比想象的要复杂得多,例如异构环境下的应用互操作问题、系统管理问题、系统安全问题等等,这些问题在集中计算模式下是不曾出现的或不突出的。传统的面向过程的技术在开发大型软件系统时已经暴露出很大的局限性,在应付复杂的分布式应用系统时更加力不从心。由此,人们自然想到了在80年代软件领域大放异彩的面向对象(OO:Object Oriented)技术。

(2) 第二代

90年代初,以面向对象技术为主要特征的第二代分布计算技术开始孕育,经过5年多的蓬勃发展,进入了成熟时期。人们也将这一代技术称为分布对象技术。

分布对象技术研究分布于网络不同节点上的对象如何进行协作,共同完成特定的任务,其核心内容在于对象之间的互操作,尤其是异构环境中的互操作问题。分布对象技术具有以下主要特点:

- 分布对象技术采用面向对象的多层客户/服务器计算模型,该模型将分布在网络上的全部资源(无论是系统层还是应用层)都按照对象的概念来组织,每个对象都有定义明晰的访问接口。创建和维护分布对象实体的应用称为服务器,按照接口访问该对象的应用称为客户。服务器中的分布对象不仅能够被访问,而且自身也可能作为其他对象的客户。因此在分布对象技术中,客户与服务器的角色划分是相对的或多层次的。
- 分布对象可存在于网络的任何地方,可被远程客户应用以方法调用的形式访问。至于分布对象是使用何种程序设计语言和编译器所创建,对客户对象来说是透明的。客户应用无须知道它所访问的分布对象在网络中的具体位置以及运行在何种操作系统上,该分布对象与客户应用可能在同一台计算机上,也可能分布在由广域网(如Internet)相连的不同计算机上。分布对象具有动态性,它们可以在网络上到处移动。
- 支持客户访问异地分布对象的核心机制称为对象请求代理(ORB:Object Request Broker)。ORB处于分布对象技术的核心位置。

与第一代的分布计算技术相比,分布对象技术的实质性进步在于使面向对象技术能够在异构的网络计算环境中得以全面、彻底和方便的实施,从而能够有效地控制系统的开发、管理和维护的复杂性。

需要特别强调的是,分布式应用系统比台式应用系统要复杂得多,这种客观存在的复杂性无法(至少很难)通过技术手段降低。人们所能够做的事情只是不要把已经很复杂的问题变得更复杂,这正是第二代技术优于第一代技术的关键。

1.2 分布式系统中的面向对象技术

在软件工程的发展史中，有两个里程碑式的突破，它们分别是70年代的结构化编程和80年代的面向对象编程。特别是近几年来，面向对象的计算已经成为软件工程领域中一项非常重要的技术，面向对象的语言（如C++，Java和Smalltalk等）越来越引起人们的广泛关注。随着面向对象技术的成熟和发展，它正在被越来越多地应用于各个计算机领域，如分布式计算、人工智能、数据库和操作系统等。本节主要探讨分布式计算系统中的面向对象技术。

分布式系统包括的范围十分广泛，它可包含任意个数的系统进程和用户进程，需要实行某种全系统范围的控制，以便提供动态的进程间的合作和运行时间的管理。与传统的集中式计算机系统比较，分布式计算系统的研究者和用户发现，后者至少具有以下优势：

（1）通过互连和互操作可以提高系统的协作能力；

（2）通过并行处理可以提高系统的性能；

（3）通过复制技术可以提高系统的可靠性和可用性；

（4）通过模块化技术可以提高系统的可伸缩性；

（5）通过动态配置和重新配置功能可以提高系统的可扩展性；

（6）通过资源共享可以提高系统的性能价格比。

基于以上原因，在最近的十多年中，分布式计算系统已经吸引了众多的研究者和用户。随着用于实现分布式计算的软件的不断出现，这一领域中的相关技术正在逐步走向成熟。但是，由于目前的分布式应用正朝着规模更大和功能更强的方向发展，并由此而导致了更复杂的结构，于是，许多研究者的注意力又集中到如何为构造分布式应用提供支持环境这一问题上。解决这一问题的关键是为分布式软件的开发提供好的工具，并为相应的分布式应用的运行提供好的环境。

采用面向对象技术来处理复杂的软件系统已经在过去的实践中得到了肯定。由于分布式系统中各实体的自治性与对象系统中对象的相对独立性有着天然的相似性，因此，人们自然想到可以用对象来表示分布式系统中的各实体。于是，研究者们开始尝试采用面向对象技术来解决大规模分布式系统中所遇到的一些问题。

在将传统的面向对象技术的思想及语言应用于分布式计算时，会遇到很多由“分布”所带来的问题，如对象定位和系统的局部出现异常等。下面将首先对传统的面向对象技术进行简单论述，然后就分布式系统对面向对象技术的影响进行详细的分析。

1.2.1 传统的面向对象技术

面向对象技术从现实世界中客观存在的事务（即对象）出发来构造软件系统，并在系统构造中尽可能利用人类的自然思维方式，使用人类在逻辑思维中经常采用的方法和原则，如抽象、分类、继承、聚合、封装等。采用面向对象方法构造系统具有对系统的理解与

客观世界一致、界面清晰、易于管理、细节变化不影响大局、易于扩充、可重用性强、维护方便、易移植等优点，因而，面向对象方法在计算机领域中得到了极为广泛的应用。

面向对象技术将客观世界中的各种事务及其间的关系都作为对象进行处理，相互关联和相互作用的所有对象构成了我们周围的客观世界。每个对象都有自己的内部结构和活动规律；多个具有相同性质的对象可以被抽象成一个对象类；复杂的对象可以由简单的对象通过某种方式组合而成；不同对象之间通过消息传递和相应服务进行关联和相互作用。

面向对象技术运用对象、类、继承、封装、聚合、消息传送、多态性等概念来构造系统。其主要特点是：

(1) 以对象作为客观世界中事务的抽象表示，并作为系统的基本构成单位。

(2) 事务的静态特征(可用数据表达的特征)用对象的属性表示，事务的动态特征(事物的行为)用对象的方法(操作，服务)表示。

(3) 对事务进行分类。把具有相同属性和服务的对象归为一类。换言之，类是对象的抽象描述，每个对象是它的类的一个实例。

(4) 用“继承”概念表达对象抽象之间的层次关系。通过在不同程度上运用抽象的原则(或多或少忽略事物之间的差异)得到较一般的类和较特殊的类。特殊类继承一般类的属性和服务。在继承中，不同类之间的公共结构与行为一般由父类表达。父类代表更为广泛的抽象，子类则是父类抽象的具体化抽象。运用继承原理对具有层次关系的类的属性和操作进行共享，可大大地减少设计和程序的重复性。

(5) 运用“抽象”的思想来处理复杂多变的系统。抽象忽略现实世界中对象和过程等实体或现象之间的差异而侧重于识别和研究其间的相似性。面向对象中的抽象代表了一个对象不同于其他对象的本质属性。抽象集中于对象的外在表现，它将对象的基本行为与其实现相分离。通过这种行为与实现的分离，对象只通过其确定的接口提供服务或行为。

(6) 封装(信息隐藏)是将对象的数据和操作合为一个独立实体，使之具有完备性和封闭性。对外屏蔽其内部细节，使对象的各种独立的外部性质与其内部实现细节相分离，从而使得系统工作人员可以对系统的各部分分别进行分析，防止由于程序的依赖性而带来的变动影响。

(7) 多态是指同一操作可以是多个不同的类的行为。多态分为两类：包含多态(Inclusion Polymorphism)和操作多态(Operation Polymorphism)。前者的意义是对父类型可用的操作，对子类型也可用；后者的意义是指没有父子关系的不同类型的对象之间可以共享某些操作的能力。

1.2.2 分布式系统对传统对象模型的影响

分布式对象已经成为众多的研究者非常重视的领域，在这一领域中已经取得的研究成果表明，很多传统的 OO 思想已不再适于正在发展中的分布式计算。例如，常规的面向对象分析(OOA)和面向对象设计(OOD)方法可以直接应用于分布式系统的分析和设计，

然而传统的面向对象编程(OOP)环境(如 C ++ 或 Smalltalk)在直接用于分布式应用系统的程序设计时遇到了问题。传统的对象与访问该对象的程序只能存在于同一进程中,并且只有相关程序设计语言的编译器才能创建这些对象并感知这些对象的存在,而外部进程无法了解和访问这些对象。这意味着在常规的分布式客户/服务器应用中,客户进程不可能直接访问异地服务进程中的常规对象。

对分布式对象的研究工作正在向传统的面向对象的原理提出各种挑战,于是,新的技术便应运而生,分布式系统对传统的对象模型产生了以下不同方面的影响:

1. 封装

封装是把对象的数据和实现服务的细节隐藏起来,客户只能通过对象的接口来访问对象的状态。封装是面向对象语言的一个重要特性,也是所有面向对象的系统所追求的一个主要目标。它所带来的优点也是分布式应用之所以采用面向对象思想的一个重要原因。

在分布式系统中,由于客户和服务器经常存在于不同的机器中,所以客户很难了解服务器的实现细节。从这个意义上可以说,分布式计算加强了面向对象的思想,它通过把客户和向客户提供服务的对象在物理上分开来阻止客户对对象的数据的直接访问。

与传统的面向对象系统比较,在分布式对象系统中,实现服务的细节更复杂,所以封装的内涵也更丰富。例如,在传统的对象语言中,假设一个对象向客户提供了两种服务——push 和 pop,客户只需申请这两种服务而不必关心实现两种服务的算法以及与算法相关的数据结构(队列或链表),也就是说,对象屏蔽了有关的算法和数据结构。但是,在分布式对象系统中,对象不仅要屏蔽以上提到的内容,更重要的是,它还需要屏蔽"系统是分布的"这一特性。

在屏蔽"系统是分布的"这一特性方面,分布式计算专家已经取得了较为成熟的研究成果。具体表现在,分布式系统平台应该向上层的应用和用户屏蔽服务的实现细节,提供以下分布透明性:位置透明性(对象在不同位置的机器上)、访问透明性(对象在不同类型的机器上)、持久透明性(对象所处的状态既可以是活动的,也可以是静止的)、重定位透明性(对象的位置已经变化)、迁移透明性(对象已经迁移到其他机器)、失败透明性(要访问的对象已经失败)、事务处理透明性(与事务处理相关的调度、监控和恢复)和复制透明性(多个对象副本之间一致性的维护)。

分布式透明性是对面向对象系统的封装特性在分布式环境中的延伸和发展,是分布式对象系统中的封装特性必须包含的内容。只有提供了分布式透明性,分布式系统中的对象机制才能达到传统的对象系统所追求的一个主要目标——屏蔽实现服务的细节。

2. 继承

对象的继承性可以提高软件的可维护性和可重用性,是公认的面向对象系统的一个主要特征。但是,在分布式系统中实现对象继承的难度却非常大。实践证明,即使能够实现,所需付出的代价也太高。所以,在分布式对象语言的设计中尽力避免继承思想,目前还不存在既支持分布式处理,又支持继承性的语言。有的文献明确指出,"继承"和"分布"不能共存。

但是,对继承性进行仔细分析后可以发现,产生问题的根源不在于继承性本身,而在

于对象完成服务时所需的连接。当一个客户请求一个对象通过继承机制才能提供的服务时,由于提供这一服务的操作代码只有一个副本,并且很有可能存在于另外一个节点上,于是,提供这一服务所付出的代价将变得非常大。如果在分布式系统中能避免这种情况发生(即消除由连接所带来的高代价),则分布式对象也可以提供继承性。这样便可以大大提高面向对象的分布式软件的可维护性和可重用性。

在面向对象的系统中,可以把对象分为两个组成部分:接口和对象实现。接口用于描述使用该对象的方法,一般来说,接口由该对象所能提供的操作的说明组成;对象实现则实际构成该对象所提供的服务,它定义了与对象有关的数据的格式和用于管理这些数据的服务。多个对象可以共享一个对象实现,其中每个对象都有自己的数据副本,而执行代码只有一份,被所有对象共享。在面向对象的语言(如 C ++ ,Smalltalk 等)中,接口和对象实现往往由一个实体(类)来表示。在开放的分布式计算环境(如 CORBA,RM-ODP 等)中,接口和对象实现往往被严格地分开,其中接口由接口定义语言 IDL 来描述,对象实现的方式则根据具体的编程语言而定。

基于以上分析,要实现分布式对象的继承,应该从接口继承出发。其基本思想如下:系统首先根据接口的继承关系生成接口的继承关系图,当客户请求某一个接口中所描述的某个服务时,如果系统检查到与该服务所对应的对象实现不存在时,可以根据接口的继承关系图,向客户返回一个能够提供该服务,并且相应的对象实现又存在的对象的访问信息。客户可以根据这些信息直接与目标对象建立连接。客户只要求目标对象能够提供自己所需的服务,而不关心是哪个对象提供了这些服务。

3. 对象引用

对象引用不仅用来在系统中标识一个对象,客户还可以根据它所知道的对象引用来与相应的对象建立连接,从而访问该对象。

在集中式的面向对象的语言中,由于所有的对象都存在于同一个地址空间中(即,所有的空间被同一个操作系统管理),对象引用往往由指向该对象所在的内存地址的指针来表示。在分布式系统中,由于对象可能分布在不同的节点上,所以就不可能像集中式的系统那样,用简单的内存地址指针来表示对象引用。分布式系统中的对象引用需要由一个较复杂的数据结构来表示,在设计这个数据结构时,要充分考虑到由系统的分布所带来的一些问题,如网络协议、网络地址和对象的迁移等。

1.2.3 实现分布式对象的机制

如上所述,分布式计算已经对传统的对象模型的概念进行了扩展。显然,若要实现分布式对象模型,就必须引入新的机制。本节将简单介绍这方面的一些基本概念,后继章节还将对此进行更深入的讨论。

1. 分布式调用

客户通过调用对象接口中的操作来获得对象所提供的服务。在集中式系统中,这种调用可由过程调用来完成,也可以通过把相应的请求信息传递给对象来完成。在分布式系统中,对象往往存在于不同的地址空间中,所以无法采用第一种方案来实现操作调用。

消息处理单元是分布式系统的核心组成部分之一，所以分布式调用可以采用第二种方法。但是，与集中式系统中的操作调用比较，分布式系统中操作调用的实现要复杂得多。这种复杂性主要来自于分布所带来的问题，即需要隐藏分布式实现的细节，实现各种各样的分布式透明性。

在一个实际的分布式对象系统中，往往不需要提供所有的分布式透明性。因为，很多分布式应用不需要某些透明性，并且，如果强求系统的大而全，常常要付出用户无法接受的代价。基于以上考虑，通常的方法是，在实现分布式调用的过程中，集中精力提供分布式应用最需要的两个透明性——位置透明性和访问透明性。有了这两种透明性后，客户就不必再关心对象在本地还是在远地，以及客户和目标对象所在的系统是同质的，还是异构的。

为获得访问透明性和位置透明性，必须在原来的对象系统中引入新的机制。在这方面已有的研究成果如下：

(1) RPC(远程过程调用)

RPC 把本地过程调用的语义扩展到分布式环境，即，它允许应用像调用本地进程一样来调用远程过程。于是，用户在开发分布式应用时，便可以不关心"系统是分布的"这一事实。RPC 利用底层的消息传输设施来为远程构造本地进程的抽象。在 RPC 中，存根(Stub)是一个关键的角色。在客户方，它把过程调用的参数和调用过程本身的名字一起封装到消息中，并把这一消息发送给接收对象；在服务器方，存根收到相应的消息后对其解封装，并且根据解封装后所得的信息来调用本地过程，最后对执行结果进行封装并将其传回客户方。存根的实现是比较复杂的，在异构的分布式环境中，它还要负责不同机器间的数据表示的转化。

(2) ORB

对象请求代理(ORB：Object Request Broker)是对象管理组织(OMG：Object Management Group)在其 CORBA 规范中引入的概念。ORB 的作用是，把客户发出的请求传给目标对象，并把目标对象的执行结果返回给发出请求的客户。因此，可以说，ORB 提供了客户和目标对象之间的交互透明性，其中包含位置透明性和访问透明性。

(3) 分布式虚拟存储器

这一技术把虚拟存储器的概念扩展到分布式环境中。分布式虚拟存储器的核心思想是，在分布式系统中建立跨越多个结点的虚拟地址空间。当访问一个对象时，首先检查该对象是否在本地(如果无法确定，则返回异常信息)。在确定了对象所在的位置后，把它装入本地的地址空间，这样，便可以使用标准的过程调用来处理分布式调用。

以上用来实现分布式调用的几种主要技术各有自己的优点和缺点。例如，如果单个客户不断地访问某个特定对象，那么，使用分布式虚拟存储器的效率就较高；如果一个对象被众多的其它对象所共享，则 RPC 方法更合适；与这两种方法比较而言，ORB 具有较高的灵活性，因为它们可以根据具体的环境做出一定的管理决策，如对象的创建、激活等。

2. 通用服务在分布式系统中的实现

同一服务在不同的对象中可以具有不同的实现，不同的实现可以产生明显不同的行为(可能有一定程度的相似)。这种具有多个实现的服务称为通用服务。如果一个系统支

持通用服务,它就可以为同一个服务提供多个不同的实现,这些实现提供看起来等价的接口。

支持通用服务的异构分布式系统可以为同一应用提供多个不同的版本,这些应用可以分别在不同的系统上运行。客户只需发出请求,系统将根据目标对象所在的系统不同而运行不同的代码。在分布式对象系统中,可以通过使用交易功能来提供通用服务。交易器通过检查客户请求的服务类型和服务器所能提供的服务的一致性来选择提供服务的对象。

1.2.4 分布式对象系统的支撑环境

在传统系统的基础上直接创建分布式对象系统是非常困难的,对一般用户来说几乎是不可能的。于是,众多的研究者和系统设计者从不同的方向对分布式对象系统的支撑环境进行了许多深入的探索,并取得了令人瞩目的成绩。

在这方面比较早的研究主要侧重于分布式操作系统和面向对象的分布式编程语言。虽然目前已经出现不少的分布式操作系统,但由于在这些系统上的应用较少,且它们难以与主流操作系统兼容,因此,这些系统并没有得到广泛的应用。面向对象的分布式编程语言(如 Smalltalk 和 Java 等)目前已基本成熟,并已被普通用户所接受。然而,仅仅利用这类语言尚不能构造比较完备的分布式对象系统,因为它们很难提供分布式系统所需的许多通用服务,如事务处理服务、时钟服务、管理服务和较好的定时服务等。

基于以上原因,中间件技术已经成为分布式对象领域中的研究热点。简单地说,中间件是处于操作系统和应用软件之间的一层软件,它在不改变现有操作系统的前提下,向分布式应用提供相应的执行环境和编程环境。由于这一技术的蓬勃发展,它已经受到国际标准化组织和工业组织的高度重视,并且已经出现了相应的标准和规范。本章的第四节将对这一技术进行详细的探讨。

1.3 Client/Server 计算

分布式系统可以和多种计算模型相容,形成不同的系统组成和结构。Client/Server(客户/服务器)模型是目前分布式计算系统广泛采用的一种计算模型。

1. Client/Server 模式的主要特点

在传统的 Client/Server 模型中,软件分为客户和服务器两部分,分别运行于不同的机器或进程中,二者协同工作。80 年代以来,一些主要的计算机技术(网络、廉价 PC、图形化用户界面和关系数据库)推动了这种模式的发展。Client/Server 模式的主要特点是:

(1) 客户/服务器(Call - Return) 工作方式;

(2) 以消息交换作为通信方式;

(3) 基于过程的服务访问;

(4) 服务集中于特定服务器。

2. Client/Server 系统的优点

Client/Server 模式的出现简化了复杂应用程序的开发和维护。Server 为 Client 提供系统定义的各种服务，如各种基于文件的服务、数据库服务、名字目录服务、事务处理等，为用户提供了一种有效的资源共享手段。与传统的分时共享模式和资源共享模式相比，Client/Server 系统具有如下优点：

(1) 优化网络利用率，减少了网络流量。客户机只把请求的内容传给服务器，服务器也只是返回最终结果，系统中不必传输整个数据文件的内容。特别是数据库操作时，由于处理数据的主要过程和数据是放在一起的，库的内容可以不必传来传去。而在资源共享模式中，由于整个文件要在本地处理，信息通常都要下载到工作站上，其间要传输大量的数据。

(2) 响应时间较短。改进的原因之一是网络的流量减少了，特别是如果当 Client/Server 模式中允许在本地留下远端数据库的副本时，数据查询的性能会得到很大的提高。

(3) 通过把应用程序同它们处理的数据隔离，可以使数据具有独立性。这样，服务器就能对数据的存取进行充分而且有效的控制，未通过鉴别和授权的用户将无法对数据进行非法访问，系统数据的完整性可以得到充分的保证。此外，数据的封装性使得改变对数据本身的操作较为容易，通过少量的改动就可把新的数据集成到已有的应用中去，可以更快地开发出新的应用。

3. Client/Server 模式的不足

传统的客户/服务器应用软件模式大都是基于“肥客户机”结构下的两层结构应用软件。客户机方软件一般由应用程序及相应的数据库连接程序组成，服务器方软件一般是某种数据库系统。它面临的一个主要问题是系统的可伸缩性差和安装维护困难。开发人员写出的程序在客户端运行，占用了大量的系统资源和网络资源。Client/Server 模式的不足之处包括：

(1) Client 与 Server 直接连接，没有中间结构来处理请求；

(2) Server 定位通常需要网络细节；

(3) Server 必须是活动的；

(4) 客户端的应用程序严格依赖于服务器端数据存储和组织方式；

(5) 应用接口的异构性严重影响系统间互操作；

(6) 许多相同的功能块被多次重复开发，代码的复用很困难。最常见的复用方式是拷贝代码块并对其进行修改。

4. Client/Server 结构模式的几个发展阶段

Client/Server 的结构模式经历了三个发展阶段，如图 1-1 所示。

在局域网阶段，早期采用文件服务器，现在应用最广泛的是 SQL 数据库服务器。SQL 最初是一种处理数据的描述型语言，随着 SQL 应用于 Client/Server 环境，人们认识到仅管理数据是不够的，还应管理处理数据的函数，为此引入了存储过程这一概念。

存储过程存在于 Server 数据库中，Client 通过类 RPC 机制调用这样的过程(如图 1-2 所示)。但 SQL 的进程管理能力很弱，且不适于管理复杂数据类型和分布于多个 Server 的数据。事务处理监控(TP Monitor)技术(在下节介绍)和群件(Groupware)技术从不同方面