计算机科学1996 Vol. 23 №. 2

分布式系统中的OO技术:OMG的CORBA

钱 方 周 健 邹 鹏
(国防科技大学计算机系 长沙 410073)

摘 要 Interoperability in Distributed System is key ponit in computer science. This article introduces a leading organization in this field——Object Management Group(OMG). The article introduces two main technologies: Common Object Request Broker Architecture(CORBA) and Object Model, and describes the applications of these technologies in Distributed System.

关键词 Interoperability, Object-orieted technology, Commercial available technology, C/S model, Remote process call.

作为开放系统的主要特征,互操作性(Interoperability)一直是计算机界关注的焦点。它可以用三个特征来评价:模块化——具有模块结构的能力;通信通用化—使用标准通信协议和接口例程的能力;数据通用化—使用标准数据表示的能力。最初,互操作性只囿于两个硬件系统之间,随着网络的发展,尤其是九十年代以来分布式系统日趋流行,大量异构性网络以及各个计算机厂家推出的各树一帜的软、硬件产品,造成了在分布式系统的各个层次上,如计算机应用、操作系统、网络系统和硬件上都广泛存在着互操作性问题。在这些层次上几乎没有关于互操作性的任何标准,已有的极少数标准(无论是事实上的,还是法规上的)只局限于低层电缆或网络层的互操作问题,现在只能用凑合的办法(例如共享文件格式和批模式格式转换程序)在异构性网络中实现用户到用户的真正互操作性。如何在计算机用户之间实现真正的互操作性,已成为计算机用户和厂家共同瞩目的焦点问题。

在分布式系统的互操作性领域,已开始了大量的研究工作,其中,处于领先地位是一个称作对象管理组(Object Management Group—OMG)的非盈利性协作组织,它遵循一个新的技术途径,即对象技术,产生了公认的基于可商用对象技术的事实标准。它的一套面向对象的标准化语言、接口和协议的基础是一种称作通用对象请求代理结构(Common Object Request Broker Architecture- CORBA)的通信约定。CORBA 在具有不同的操作系统、语言、网络协议和硬件结构的系统间提供了应用层的互操作性,实现了从用户到用户的互操作性。OMG 未来的计划是运用这些有力的服务和功能支持分布式环境上应用程序员的设计开发。

一、对象管理结构(OMA)

OMG 的宗旨就是产生一套标准的语言、接口和协议,以支持异构应用间的互操作性。与试图在计算机其它层次(硬件、操作系统、应用接口)上建立同构的组织不同,OMG 处理计算环境长期存在的异构性问题(迫于竞争和需求的压力)。OMG 的一套接口建立在现存的接口之上,而不是代替它们。

实现互操作性的关键技术是面向对象技术。由于面向对象技术提供了结构良好的模块、精确定义的接口和现有软件的可重用(继承)等先进的机制,它是在多机种环境中定义服务的最佳选择。而且,OMG 一开始就利用了商业可行性技术,并因此产生了与流行的产品兼容的广阔市场。这两个关键技术——面向对象技术和商业可行性技术构成了 OMG 发展的基石。

OMG 基于面向对象技术和商业可行性技术,采用自底向上的方式,从网络开始,向上一直到应用接口形成事实上的标准。OMG 所采用标准的第一步就是对术语和结构进行约定。这个方面有关的方向、结构和对象的早期约定是从1990年开始的,在

(对象管理结构导论)[OMAG]中发表。它首先给出一种参考模型,对象管理结构(Object Management Architecture),即在一个结构化方式下选择技术的基础。然后,基于这一模型广泛征集信息和建议,从OMG成员中获得关于现有产品的详细技术和商业可行性信息,用来充实参考模式结构中的不同部分。在技术和商用委员会审查了这些反馈信息后,OMG的决策者将最终决定采用哪种技术。被采用的约定无论对组内成员还是非组内成员来说都是方便可用的。以下就是这一参考模型:

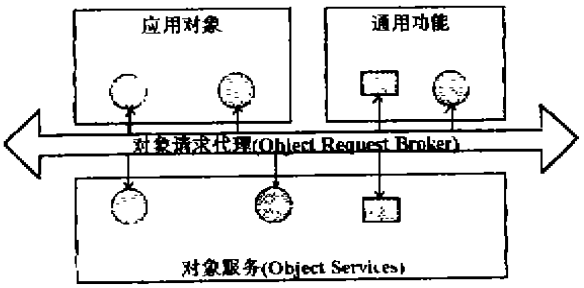

图1 对象管理结构

在OMA模式中,每一个软件代表一个对象,它们通过对象请求代理与其它对象通信。对象请求代理(ORB)是一个关键的通信机制,在高互操作性方式下,处理应用对象之间的消息分布。基于标准化过程的相对重要性,OMG将对象分成了三个大类:对象服务、通用功能和应用对象。其中,对象服务提供实现基本对象功能的主要函数,它使用对象请求代理处理对象的逻辑模式和对象的物理存贮,如保密和认证、接口转换等,这些服务的实现是低层的;通用功能包括在许多应用领域都有用的功能,通过OMA所允许的类而实现(如电子邮件和其它应用层)的服务等,这些服务的实现是高层的;在这一模式中所标明的应用功能是一片属于应用开发者的广阔天地(无论是独立的软件卖主,还是增值转卖商)。OMG在这一领域中将不规定任何标准。

要充实对象管理结构(OMA)这一参考模式,必须明确四个标准化领域:对象请求代理(ORB)、对象模式、对象服务和通用功能。其中,对象模式是针对应用对象提出的,即必须把应用对象设计成可移植的抽象模式,可以和OMG兼容的面向对象系统通信。

OMG最近的决策过程导致了两种主要技术的出现:对象请求代理层(即通用对象请求代理结构,CORBA)的基本通信(消息)代理,和标准的、可扩充的核心对象模式,而对象服务和通用功能的选择还在酝酿之中。

二、通用对象请求代理结构(CORBA)

OMG在分布式系统中互操作性技术的最先实现体现在它所采用的技术规范——CORBA中。它借助于面向对象技术,有效地实现了通信、语言和应用上的互操作性。

(1)C/S结构。按照面向对象的观点,计算机系统的资源是封装在各个管理对象中的,而一个计算机系统常被认为是管理和管理对象的集合。那么,如何访问这些对象,获取系统管理资源呢?作为当今计算机界三大主流技术之一的Client/Server结构,既能有效地实现资源的管理,又能充分体现面向对象的思想,所以被OMG纳入CORBA中。

CORBA(或任何调用分布式服务的)所提供的基本服务,是一段代码到另一段代码之间消息(请求)和应答(返回或异常)的传递,通常称这两个传送请求与应答的对象分别为客户和服务器。虽然在CORBA中,任一对象可以或者是客户,或者是服务器,或者两者都是,但事实上,我们期望大多数的对象(如应用)在它们的生命周期中既是客户又是服务器。按照面向对象的观点,CORBA中的服务器都是由对象实现的,所以CORBA中称它们为对象实现。

这样,传输媒介的功能就是提供客户和服务器通信的不可视性传输。通常有三种通信途径:对话(Conversation)、消息排队和远程过程调用(RPC)。下面以开放软件基金会(OSF)/分布式计算环境(DCE)的RPC为例:

CORBA对RPC做了一些改动从而简化了程序员的工作,也降低了维护费用,提高了产品的可扩充性。

CORBA编译的分布式系统用两种完全不同但兼容的方式执行RPC的功能:

·在动态机制中,客户可以在运行时间内搜寻可用的服务器(例如,基于地址和功能),找到这些服务器

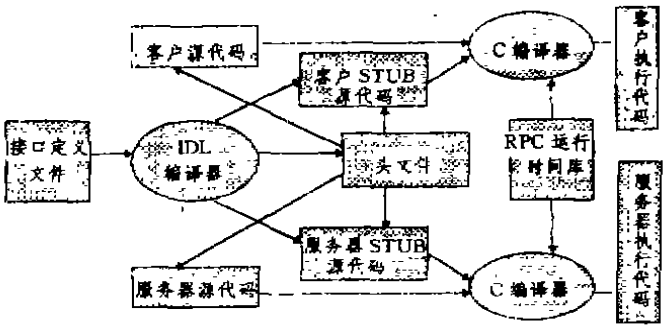

图2 DCE 的远程过程调用(RPC)

的接口、构造使用这些服务器的请求、并发送这一请求(任意多次)、最后接收应答。上述过程即使在编译时不知道可用的服务器和接口信息、也是能够实现的;

·在静态机制中、用户必须在编译时就知道要访问的服务器的接口、虽然在运行时间内才能找到可用服务器。

为什么要有两种机制呢?很简单.使用静态机制是有充分的工程原因的,虽然对某些应用来讲动态机制是相当方便的(有时甚至是必须的),但很明显(至少理论上是这样),静态机制有较好的性能,因为在编译时就必须知道实现语言、操作系统和通信信道的所有细节,所以能采用各种优化技术降低使用通信信道的执行时间耗费。

但不管怎样,有些重要的应用是非用动态消息机制不可的,最典型的例子是浏览器:在一个分布式环境中为动态跟踪调试要传递一个浏览器,而这在静态机制中几乎是不可能的,因为在运行时会加入新的服务器和接口。

CORBA 既采用了动态机制又采用了静态机制,是通过接口定义语言实现的、相同的语言用于编译、生成 Stub 和 Skeleton(仅仅在 RPC 中),能使用户在运行时间找到接口、并构造请求。

这样静态、动态两种接口的选择,使客户和服务器的程序员能基于要开发软件的需求,选择其中任意一种。更有利的是,它甚至允许应用软件在同一段代码中同时使用静态、动态两种机制。当程序员在编译时知道了已定义好的接口时、就可以使用静态机制,而对其它的请求,可以使用动态机制。其中,接口池(IR)是所有可用接口的动态表示、标明了在分布计算系统中所有可用对象的接口(或类)(图3)。

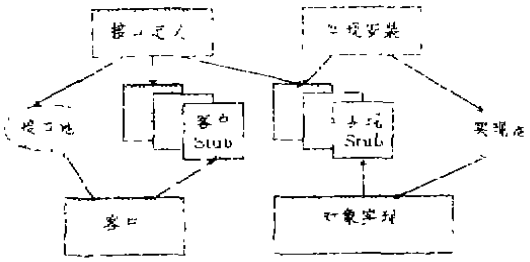

图3 接口定义语言

比较DCE和CORBA的RPC有下述三点不同:第一、如上所述,DCE 的 RPC 只是一种静态调用机制、必须在编译时明确调用接口、而在 CORBA 中,提供了动态、静态两种调用机制、程序员可以根据具体情况灵活选择任一种调用机制;第二、CORBA 提供了调用的非间接性、即客户和服务器之间没有直接的调用关系、而是通过 ORB 来传递请求、客户和服务器之间的一个请求要通过一个或多个 ORB 的传递而实现;第三、CORBA 环境中的对象粒度要比 RPC 服务器中的小些,这样分布式环境中的一个应用才可以方便地使用其它对象上的服务。

综合上述、CORBA 和 DCE 的 RPC 最本质的不同在于,DCE 的 RPC 未引入面向对象的思想、没有提供实现客户和服务器之间互操作性的技术。而 CORBA 在客户和服务器之间提供了两层互操作性:首先、它将客户和服务器都抽象成对象,不必关心对象的具体实现,所有的功能都封装在对象内部,而提供给其它对象的是简单的接口,这些接口被放入接口池中,可以被其它对象以动态或静态的方式调用;其次、客户和服务器之间通信是通过 ORB 这一代理实现的、对象不必关心通信的细节,由 ORB 定址合适的对象,并发送请求。

(2)接口定义语言(IDL)、在 CORBA 中有唯一的说明语言,称作 OMGIDL(即 OMG 接口定义语言),用以说明独立于执行(可编程)语言的接口。虽然 IDL 本身不是一个可编程语言,但它为程序员提供了语言的独立性,他们不必知道调用者所采用的语言。IDL 也是面向对象的,允许接口表示的抽象

(封装)、多消息(功能调用)和接口的继承。IDL 实现了语言层次上的互操作性。

(3)对象适配器(Object Adapter)。除了在客户、服务器以及 ORB 之间提供静态、动态两种接口外，CORBA 还提供了通信基层管理自身的类属 ORB 接口。ORB 核心通信基层的重要扩展称作对象适配器，它可适用于对象的类属 ORB 模式，以及在不同对象中的实现细节。例如，对象可以用操作系统进程中的应用代码实现，也可以表示为面向对象数据库的对象，或者为在单一地址空间中通信的程序语言层次的对象(如 C++ 或 SMALLTALK)。对象适配器从消息基层中隐去了对象实现的细节。

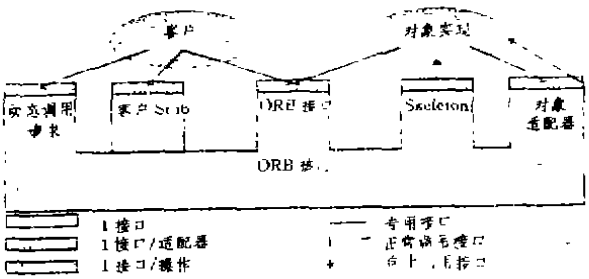

图4 ORB 接口调用

三、对象模式

事实上，在实现 CORBA 标准中尚未发现的最大困难是，竞争厂商们采用的对象实现方法的不同。使用对象的每一个人，都对对象代表/表示什么，一个对象有多大(例如，以字节为单位)，以及对象操作的频率，有不同的观点，例如，C++ 程序语言中的对象，与通常意义上的管理对象几乎不相同，只是抽象上有些相似而已，因此 CORBA 约定还包括一个单独的对象模式，在接口定义语言中得以体现，CORBA 对实现就是将这一类属模式映象到客户和服务器所用的实现(语言)中。

CORBA 模式为用户提供了一个丰富的可扩充类型集合，包括：16位和32位有符号整数、布尔值、8位不透明的数据类型(八位字节)、枚举类型和通用类型(任选)。而且，还可用几种构造类型扩充这一类型集合：有序对(名字、类型)的记录、类型的可区分联合、变长或定长的类型序列、固定长度的类型数组、说明类型操作的接口。而且，还有一个对象引用(或句柄)，可以用来传递请求，或作为通用参数。

仅仅承认这样一个类型集合当然是不够的：CORBA 标准还包括一个应用程序接口(API)。API 在 IDL 中定义，因此，它的实现也可以映象到用户所使用的任一种语言中，并且通过这一 API 接口所访问的不同结构，应尽可能地用对象来表示，使用户所使用的接口连续、简单。CORBA API 包括五种基本接口：同步动态启动请求和应答请求、延迟动态启动请求和应答请求、存贮管理和内务处理请求、上下文(基于名字参数)列表维护、接口池搜索请求，通过一个 ORB 静态请求的启动完全是在编译过程中实现的，因此它不在 CORBA 的 API 中出现。

四、CORBA 的应用

目前大约有100多个公司(包括 IBM，Hewlett-Packward，Olivetti，NEC，Sun 微软公司和许多小厂家)和十多个工业合作伙伴(包括开放软件基金会，UNIX 国际，国际多媒体联合会，X/OPEN 等等)表示要支持 CORBA 标准，并且在最近的产品中都实现了 CORBA 技术。

尤其是开放软件基金会(OSF)在它的分布式管理环境(DME)中也纳入了 CORBA 技术，DME 是一个关于分布式系统管理的范例，它将分布式的系统管理分为网络管理、系统管理和分布式系统三类，在系统管理中它建立了对象管理框架(OMF)，OMF 是 CORBA 的一个实现，其中，ORB 被更名为管理请求代理(MRB)，但功能是一样的。在 OMF 中，MRB 是运行在每个系统上的一个过程，任何一个对象本身就是一个过程，可以与本地的 MRB 通信，向其它对象发送请求。如果请求的对象在另一个系统上，MRB 会将请求传送给那一个系统的 MRB，MRB 之间是通过 DCE 的 RPC 通信的。一个典型的 OMF 一样会包括散布在一个或多个系统上的一定数量的对象(图5)。 (下转第83页)

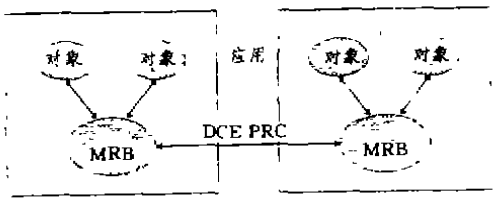

图5 一个 OMF 的应用

CBox 对象实现。异步方法调用总是立即返回一个 CBox 对象。当需要返回结果时，发送者向 CBox 对象发送求值消息获得返回值。如果结果尚未返回，则发送者等待。

5 COOL，由斯坦福大学开发，是对 C++ 语言的扩充，允许在共享存储环境中进行中粗粒度并行。COOL 允许三层并行性：对象内并行（每个方法都可以异步执行）、对象间并行（不同对象的方法可并行执行）和方法内并行（同一方法内的不同函数调用可并行执行）。

对象的方法既可以是异步的，又可以是互斥的。互斥方法可以实现函数级同步，并且比 monitor 更灵活，允许更大的并行性。

COOL 提供未来变量、Spin Lock 和 Blocking Lock 以实现细粒度同步。和 POOL-T 一样，COOL 不支持继承、友元函数（Friend Function），虚函数（Virtual Function）和方法重载。

6 ABCL/1（Actor Based Concurrent Language），基于 Actors 模型，对象执行脚本（Script）。脚本定义了对象行为、可响应的消息集和响应方式。独立对象可以并行执行。但在对象内部，消息被顺序处理。对象在创建时处于休眠状态（Dorment），直到被消息唤醒。对象可以用一个脚本模式选择所响应的消息，并把当前不能响应的消息放在消息队列尾，供以后处理。处理完消息后，对象执行一个 Select 结构，回到休眠状态。

通常消息传递是异步的，点到点的。消息中含有消息标志、参数、发送者名和返回地址。对象有两个消息队列：普通消息和快件消息（Express Messege）队列。快件消息可以中断普通消息处理。

ABCL/1支持三种消息传递：过去型（Past），现在型（Now）和未来型（Future）。过去型和未来型消息传递是异步的，现在型是同步的。ABCL/1也支持广播通信。

四、结论

对并行面向对象语言的研究已进行了很多年，并产生了大量的新语言，但还没有一个得到真正广泛的应用。究其主要原因是对象机制与并行机制的结合，远比想象的要困难。根据 Bertrand Meyer 对纯面向对象语言的定义：(1)模块结构；(2)数据抽象（对象是抽象数据类型的实现）；(3)自动存储管理（系统自动回收无用对象）；(4)类（所有的非简单类型都是类）；(5)继承；(6)多态和动态联编；(7)多重继承和重复继承。目前没有一种并行面向对象语言满足所有这七条要求。并行面向对象语言距离实用化，还有很长的一段路要走。（参考文献共15篇略）

（上接第79页）

五、结束语

虽然 CORBA1.1 文档定义了足够多的 IDL 语言和 API，但它并未考虑从 ORB 到 ORB 的通信标准。例如在 DME 中只能用 DCE 的 RPC 完成 ORB 之间的通信。为了弥补这一不足，OMG 最近正致力于开发（到1994年初）一个向上兼容的2.0修改版本，以期实现从 ORB 到 ORB 的通信标准。这样，任何两个 CORBA 的 ORB 实现，即便是独立实现的，仍可保证有互操作性，而且可共享名字域。其它问题，象本地事务处理、保密协议、服务质量控制和多重预测扩充，也将是 ORB 下一版本所追求的目标。象任何一种技术一样，CORBA 将没有止境。

参考文献

[1]Richard Mark Soley, Role of Object Technology in Distributed Systems, OMG, Inc. 1994

[2]David Chappell, The OSF Distributed Management Enviroment, Chappell and Associates, 1994