分布式系统中对象获取方法的研究*)

The Study of Method of Get Remote Object in Distributed System

王汝传　陈云芳　屠　翊
(南京邮电学院计算机科学与技术系　南京210003)

Abstract With the development of the network, the application of distributed systems becomes more and more. DCOM and CORBA are very popular technologies currently. DCOM provides a standard using component model to develop distributed systems. CORBA defines true interoperability between objects by specifying how ORBs from different vendors can interoperate. In this paper, we will discuss how to get remote objects and their character in DCOM and CORBA Architecture.

Keywords DCOM, CORBA, Object oriented, Distributed system, RPC, ORB

1　引言

DCOM是微软公司提出的一种分布式组件对象模型(Distributed Component Object Model)。DCOM起源于动态数据交换(DDE)技术，实现应用程序之间共享数据的动态交换，对象连接与嵌入OLE就是从DDE引伸而来。随后引入的组件对象模型(Component Object Model)，形成了对象之间实现互操作的二进制标准。DCOM是COM在分布计算方面的自然延续，它为分布在网络不同节点的COM构件提供了互操作的基础结构。

CORBA(Common Object Request Broker Architecture)是对象管理组织(OMG)为解决分布式处理环境中，硬件和软件系统的互连而提出的一种解决方案。CORBA1.1版本定义了接口定义语言IDL(Interface Definition Language)，开发出对象请求代理ORB中间件，在客户机/服务器结构中，ORB通过应用程序接口，实现对象之间的交互；CORBA2.0版本提出了IIOP(Internet Inter Object Protocol)，用以规范不同厂家的ORB之间的真正互通。使用CORBA，用户能在不知道软件和硬件平台以及网络位置的情况下透明地获取信息，CORBA为当代计算环境带来了真正意义上的互联。

这两种系统解决方案是面向对象技术和分布式技术结合的产物，同时伴随着网络的迅猛发展而得到广泛应用。软件开发者之间的对象调用和共享将极大地改变网络软件的生产模式，由此也必将带来软件生产技术上的革命。

2　远程过程调用和远程对象调用

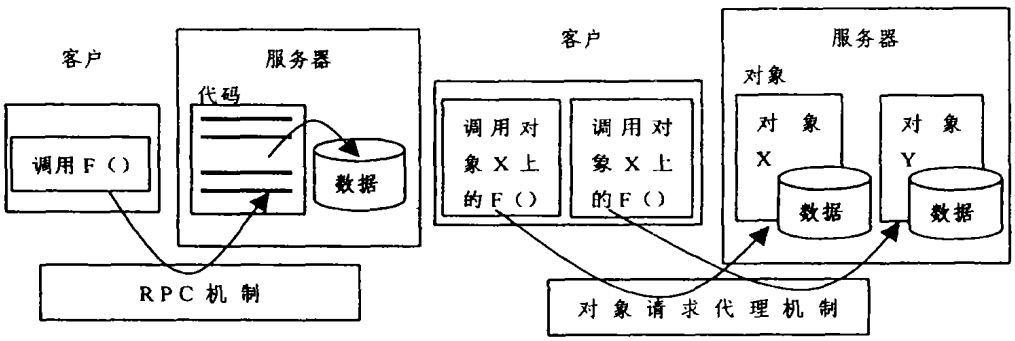

图1　远程过程调用和远程对象调用

如果说远程过程调用是面向过程的程序设计方法在网络中的应用的，那远程对象请求技术就是面向对象程序设计方法在网络中的应用。它们的机制相同，但是有许多重要的差别，远程过程调用可以调用到某个特定的功能，过程的代码和数据是分开的。而对象请求代理调用的是位于某个特定对象中的某个方法。由于对象的多态性，不同的对象类对同一个方法调用的响应是不同的，因为各个对象管理着各自的实例数据。因而方法是在某个具体的实例数据上实现的。对象请求调用方法可以作用到控制特定数据的特定对象上，然后按照类的特定方式执行功能，相反，远程过程调用则没有唯一性，所有具有相同名字的功能以相同的方式执行，无法区分具体的服务。

尽管远程过程调用在调用性能、配置难易度等方面有优势，可以迅速开发，高效工作，在小环境里十分适合，但是不会在企业系统上有所作为。在建立企业级应用时，系统的可扩展性、可靠性、可伸缩性等质量要素显得更加重要。使用远程对象请求技术是实现这些性能的基础，它能解决远程过程调用所不能解决的一些问题。

① 互操作性：良好的平台无关性应该包括程序设计语言的无关性，即用一种语言编写的客户程序可操作由另一种语

*)本文得到国家自然科学基金(60173037)和江苏省自然科学基金(BK2001123)资助。王汝传　教授，博士生导师，主要研究方向是计算机软件理论、计算机网络等。陈云芳　研究生，主要研究方向是计算机软件和分布式对象技术。屠　翊　研究生，主要研究方向是计算机软件在网络中的应用。

言编写的服务对象。要实现对象之间的可互操作性，开发者必须使用统一的接口定义语言来描述所有服务接口，而服务的具体实现可选用不同的程序设计语言。

② 可靠性：分布式系统要求更高的可靠性，客户程序可能由于各种原因导致无法访问远程对象。理想的情况是能在一个服务程序失败时自动转向另一个服务程序，甚至能自动激活服务程序后再调用其中的服务对象，允许客户程序透明地重新绑定到服务对象。

③ 对象命名与查找：实际的企业级软件系统中存在大量服务，如果所有客户程序都使用简单的远程过程调用来解释所需服务，无疑会带来很大的混乱。传统的方法是通过作用域规则在不同层次的名字空间中命名与查找对象，在分布式软件系统中需要分布式的对象命名与查找机制。

④ 对象策略：企业级应用在客户请求增多时会引起请求的排队与阻塞，有必要激活更多的对象实例提供服务。远程对象代理可根据客户请求自动激活新的伺服对象实例，也可以由系统管理员手工激活与冻结伺服对象实例，最好能由分布式软件开发环境与运行环境提供一种显式的支持。

以上几个企业级应用所必需的功能都是远程过程调用所无法实现的，只有实现远程的对象代理机制，才能有效地解决这些问题。

3 远程对象的获取

3.1 DCOM 中的实现方法

DCOM 是建立在 COM 的基础之上的，首先我们来看一下 COM 是如何获取对象的。COM 对象的管理主要是通过 Windows 系统注册表来实现的。一个 COM 对象要能够被其它程序所使用，它首先必须在系统注册表中进行注册。客户程序通过系统注册表来查询所需要的 COM 对象，获取相关文件路径、服务类型、版本信息等。有三种对象可以作为服务器对象，进程内服务器，本地进程外服务器，远端进程外服务器。在具体的程序编码实现时，客户端程序调用接口创建的相关函数，给出被请求对象的类标识符和接口标识符，被调用函数将把接口的指针返回给客户程序。客户程序可以是任何包含指定的服务器对象函数指针的代码，一旦获得了接口的指针之后，它能通过这些指针来调用接口的具体实现代码。COM 对象是通过全局唯一标识符 GUID(Globally Unique Identifier)实现对象的识别的，操作系统和其它软件通过这个标识符来使用所需的 COM 对象，标识符的产生就代表了该接口的发布。注册的 COM 对象的信息都放在系统注册表 HKEY_CLASSES_ROOT\CLSID 目录下。

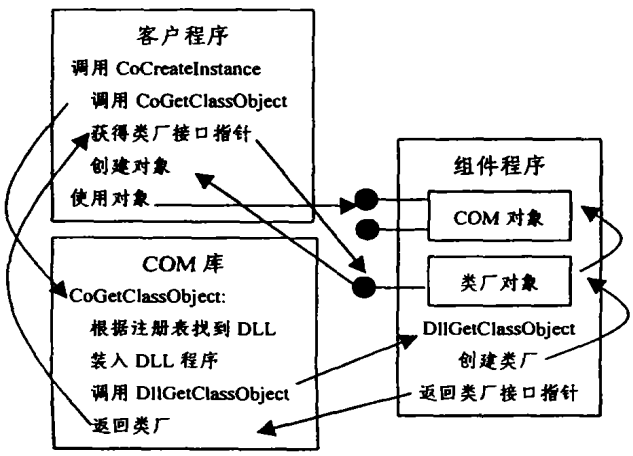

图2 COM 对象创建过程

COM 对象的创建过程如下：客户程序调用 COM 库中的 CoGetClassObject 函数，根据注册表中的信息找到相关的 Dll 文件，装入 Dll 文件，调用 DllGetClassObject 函数在组件程序中创建类厂对象，并将类厂对象的接口指针通过 COM 库传回到客户程序，客户程序获得类厂对象之后，再在组件程序中通过该类厂创建实际的 COM 对象，最后将 COM 对象的接口指针返回给客户程序，客户程序就可以通过该接口指针来调用相关的函数功能。

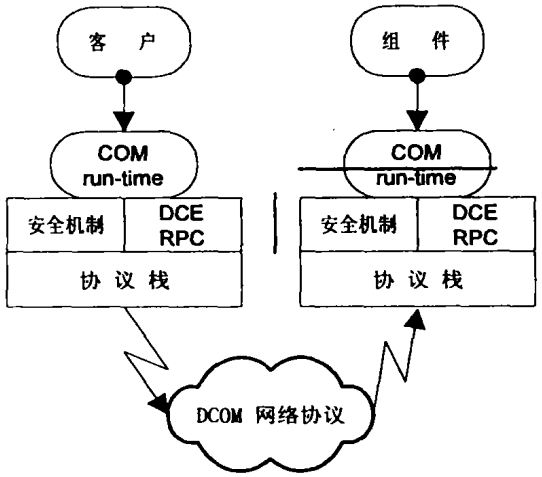

图3 DOCM 结构

DCOM 把 COM 模式推广到分布式环境中。它是基于分布式计算环境(DCE)下的远端过程调用。从本质上讲，DCOM 中对象的创建过程和 COM 中的对象创建过程基本一致，DCOM 仅仅只是用网络协议来代替本地进程之间的通讯。DCOM 依赖 ORPC 来完成它所有的网络通信工作。

DCOM 中连接到远程对象所需要的信息都包含在一种叫做字符串绑定(string binding)的特殊类型的字符串里。字符串绑定是一个 Unicode 字符串，它包含诸如机器网络地址、网络协议的令牌、通信端口等信息，用于进一步缩小和识别 RPC 目标。

在指定的远程机器上每一个运行的 COM 服务器都有一个相关的对象输出标识符(OXID)，通过给定的 OXID，我们就可以查找访问指定的服务器所需的字符串绑定，OXID 是获得字符串绑定的关键。而 OXID 的管理是通过解析器对象来实现的。

OXID 表包含一个或多个对象的服务器，当我们获得 OXID 表之后，还需要对表中的信息做相关的处理来获得正确的对象实例，这个过程就是通过对象代理和接口存根来实现的。DCOM 采用网络数据表示法(NDR：Network data representation) 打包来实现异构网络中的数据传输。进行传输的实际上是一个叫做 OBJREF 的数据结构，它除了包含一个 OXID 表之外，还包含关于在远程对象的机器上的 OXID 解析器的字符串绑定。

在 COM＋中，所有组件的信息不再存放在系统注册表中，而是单独放在一个 COM＋目录中的系统数据库中，便于对象的命名和查找等管理。该目录通过一个系统提供的实用对象层次结构进行管理，层次结构中的所有对象都支持脚本操作和远程操作。

3.2 CORBA 中的实现方法

首先介绍一下对象引用的概念，对象引用包含对象的

IDL 类型和其它的一些相关信息，这些信息可以让 ORB 找到其主机上的目标对象，当客户机得到对象引用后，ORB 将引用转化成本地编程语言对象代理，客户机应用程序用它来实现远程目标对象的激发。

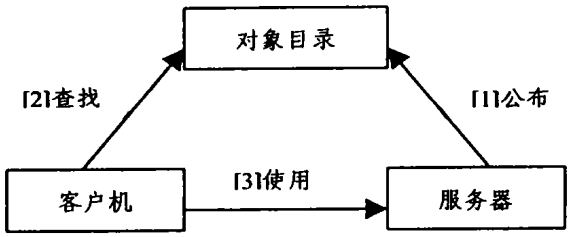

图4　对象定位抽象模型

客户机和服务器是用户所使用的应用程序组件，他们分别使用和实现业务对象，对象目录负责存储对象引用和一些描述数据。在定位对象时，服务器首先把许多对象公布到目录，提供一些有意义的方式来识别属性。客户机通过提供相关的属性在目录中查找对象，客户从目录中得到这些对象引用后就可以使用这些对象。

在 CORBA 中，获取对象最简单的方法是使用对象引用字符串，使用 object_to_string()方法，该方法把提供的对象引用字符串化，把它转化为标准的字符串格式，客户机应用程序激发 string_to_object ()方法，通过 ORB 把该字符串再转化为对象引用。

服务器通过把对象字符串化来公布它的对象，从逻辑上讲，这些字符串化的对象引用存储在对象目录中，而实际上，它们会存储在客户端配置文件或外壳脚本中。客户机应用程序通过从这些文件中获取字符串，并把它们转化为对象引用来查找对象。客户机和服务器的程序员必须协调好这些对象引用字符串如何管理，当服务器下一个文件写入对象引用时，客户机开发者必须知道这个文件的名字以及里面所含的引用字符是什么对象，然后才能正确地使用这些字符串。

在 CORBA 系统中另一个模式是使用工厂对象。该对象的一个方法激活的结果是返回另一个对象引用。工厂对象在客户机需要使用大量对象的时候极其有用。服务器可以只公布一些工厂对象，而不公布所有伺服对象的引用，客户机可以使用这些工厂对象来获取它所需的剩余对象的引用。工厂对象对于减少某个时间在服务器进程中激活的对象数量十分有效。工厂对象可以在客户机明确提出请求对象要求时再实例化该对象。工厂对象对于多数大规模的 CORBA 系统都是很重要的。

为了在更加复杂的环境中实现对象定位，在 CORBA 公共对象服务规范中定义了两种服务——命名服务(Naming Service)和交易服务(Trading Object Service)。这两个服务都可以把一些附加信息和公布的对象引用关联起来，并把这些信息提供给用户。把特定应用程序信息依附到对象引用的能力是这两个服务的主要价值。它允许用户在对客户机程序有意义信息的基础上定位对象。

CORBA 命名服务存储每个对象引用的名字，它可以提供层次化的命名空间结构，以对业务领域有意义的方式来有条理地组织对象。在类似于文件系统的层次结构中存储对象引用，每个对象引用都有一个相关联的名字。该名字包含两个字符串域，id 和 kind。分别相当于文件系统中的文件名和扩展名。层次由相当于文件系统中目录的命名上下文组成。在同一个命名上下文下存储的多个对象引用的 id 和 kind 域必须有一个不同。

实际使用命名服务时，应用程序是通过激发命名上下文对象来实现的，具体是由命名上下文所支持的 resolve 和 bind 操作来进行对象定位。

函数原型为：void bind (in Name n , in Object obj) ;
Object resolve (in Name n) ;

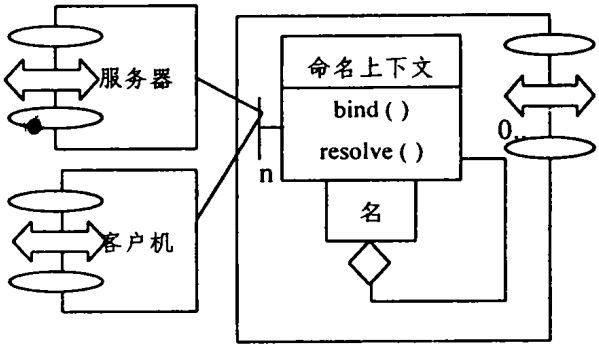

图5　命名服务

服务器调用命名上下文的 bind()方法来公布对象，提供和该对象关联的 Name 结构和对象本身作为参数。Name 结构包含两部分识别信息，其一是名字本身，由 id 和 kind 组成；其二是对象在命名服务层次中的位置。客户机调用命名上下文的 resolve()方法来查找对象，客户把所要求的名字作为参数传递进来，而被调用的命名上下文决定服务在层次中的哪个位置执行查找。如果找到匹配的对象，它就作为通用的 Object 对象返回。

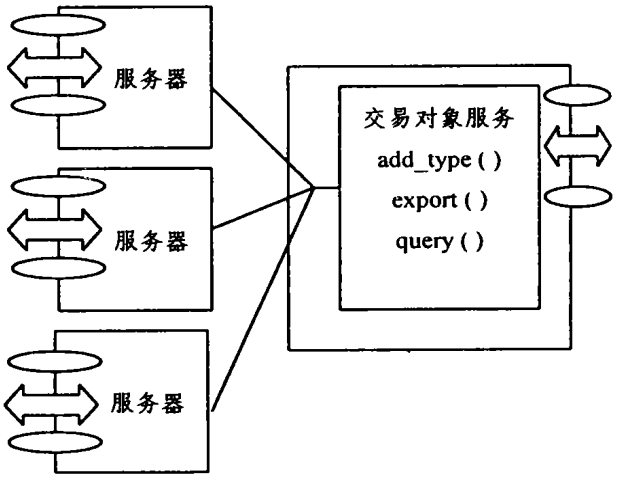

图6　交易对象服务

CORBA 交易对象服务提供了公布和查找对象的功能强大而又灵活的方式。服务器公布对象时，它们把任意类型的任意数量的属性和对象关联，客户机执行查找时，明确提出一系列所要求的属性，交易者评估该寻找查询，并返回一系列匹配对象。交易者的对象目录不以任何形式化的方式来组织。它是基于服务类型的概念，一个服务类型包含一个 IDL 接口标识符和一些定义和该类型关联属性的附加数据。

交易对象服务的 IDL 描述比较复杂，这里只做抽象的说明。首先我们必须定义所支持的服务类型，定义对应的 IDL 接口和描述这个服务供应的属性集合，定义了服务类型之后，服务器就可以公布该类型的对象，由服务器向交易者输出的

供应集合组成了它的交易空间。最后，客户机通过执行查询在交易空间中查找对象。

以上方法复杂度不同，在应用时主要考虑的是需要公布对象的数量，客户机查找对象的标准，用户需要的服务质量等等因素。对于很简单的静态环境，使用对象引用字符串是合适的，因为它不需要任何的附加服务，客户机只需要使用对象引用字符串就可以连接到合适的服务器。对于复杂环境下，命名服务和交易服务是优先选择，它们有细小的差别。命名服务有固定的多维层次结构，需要绑定的每个对象只有它的名字和在层次中的位置两部分关联信息，能够实现高效的查询。但是命名服务只能返回所需的单独一个对象，而且在命名层次发生变化时会对客户机应用程序影响较大。相比较而言，交易服务是二维的，每个服务类型可以有任意数量的属性，服务类型的增加不会影响已经存在的供应。在客户机使用变化的标准来查找对象的时候，交易服务就是更好的选择。

结论 本文主要在对远程过程调用和远程对象调用做了简单对比的基础上，对在 DCOM 和 CORBA 中获取远程对象的方法进行分析和研究。CORBA 和 DCOM 中接口的概念是一致的，都是把客户与组件分离，把接口与实现分离，组件与其客户之间通过细粒度的接口进行交互。CORBA 中各种语言的映射机制实现其语言的无关性，DCOM 中的语言无关性建立在面向对象程序设计中的纯虚函数概念的基础上，凡是符合该规范的语言都可以实现 COM 组件。

它们都提出对象工厂的概念，通过对象工厂来找到或者建立所需的远程对象。CORBA 中的对象工厂和 COM 中的类工厂相似，用来创建一个我们实际使用的对象。COM 中的类工厂只能找到特定的对象引用，而 CORBA 中的对象工厂还可以返回其它的对象，主要用来返回已存在的大量的对象引用。

它们都有一定的命名解析机制，在 CORBA 中使用较多的是命名服务，通过层次化的命名空间来解决命名冲突和实现对象的有效管理。在 DCOM 中相对应的是解析器的概念，它负责管理对象输出标识符表格。在传输远程对象所需要的参数的机制中，它们都使用打包(marshal)机制，只是具体的打包方法不同。

可以预见，本文中所论述的方法是未来分布式计算和分布式系统中获取远程对象的必然选择。

参 考 文 献

1 Object Management Group. Common Object Request Broker: Architecture and Specification, 2001

2 Object Management Group. CORBA services: Common Object Services Specification, 1998

3 Vinoski. New Features for CORBA 3.0. Communications of ACM, 1998, 41(10)

4 Schmidt D C. The Design and Performance of Real-Time Object Request Brokers. Computer Communication, 1998, 21(4): 294～324

5 潘爱民. COM 原理与应用. 北京：清华大学出版社，1999

6 Slama D, garbis J, russell P. enterprise CORBA. Prentice Hall, 1999

(上接第145页)

```
    // Input Elements for Winsock
    WSABUF m_wsaInBuffer; // WSARecv 使用的接收缓冲
    BYTE m_byInBuffer[8192];
    // Output elements for Winsock
    WSABUFm_wsaOutBuffer; // WSASend 使用的发送缓冲
    CRITICAL_SECTION m_csWriteBuffer; // 保护写缓冲的临界量
};
```

3.3 工作线程

工作线程的流程图如图3所示。

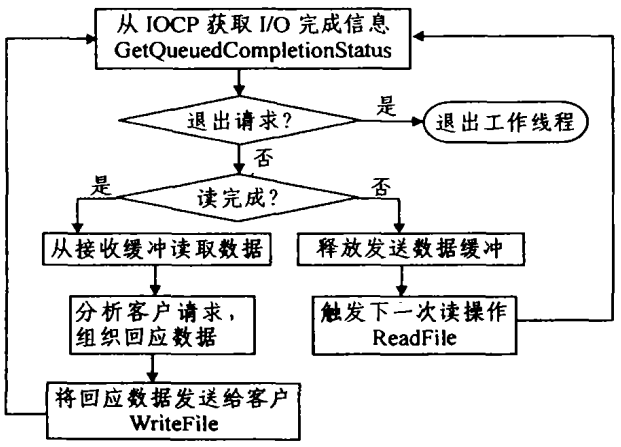

图3 工作线程流程图

一旦完成端口已创建并与 sockets 进行关联，一个或多个工作线程就可以进行完成通知的处理。工作线程调用 GetQueuedCompletionStatus 函数从 I/O 完成端口队列取得完成通知，从其第三个参数 lpCompletionKey 指向的客户上下文信息中获取与 socket 相关的上下文信息，判断其完成通知状态，进行相应的业务处理。如果发现完成信息包是退出指示，则结束线程。

```
BOOL GetQueuedCompletionStatus(
  HANDLE CompletionPort // handle to completion port
  LPDWORD lpNumberOfBytes, // bytes transferred
  PULONG_PTR lpCompletionKey, // file completion key
  LPOVERLAPPED * lpOverlapped, // buffer
  DWORD dwMilliseconds // optional timeout value
);
```

结论 基于 I/O 完成端口技术，服务器应用可以得到很好的可扩展性。本文比较了服务器应用对客户连接各种处理方式的优劣，描述了 IOCP 的工作原理，并在此基础上较为全面地分析和介绍了实现 IOCP 的各个步骤，并且详细描述了其中关键函数的用法。

微软在 Windows NT 和 Windows 2000上提供了 IOCP，并且在 IIS 中采用这种技术处理客户请求，得到了很好的效果。可以预见，将会有越来越多的服务器应用采用 IOCP 技术。

参 考 文 献

1 Reilly D J. 基于服务器的应用程序内幕. Microsoft Press，北京大学出版社，2000.9

2 Richard J. Windows NT 核心技术. 清华大学出版社