第27卷 第1期 电子科技大学学报 Vol.27 No.1
1998年2月 Journal of UEST of China Feb. 1998

# 面向对象的互操作技术*

苏 森** 唐雪飞 刘锦德
(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】 面向对象的互操作技术是信息技术领域内未来几年中的一个重要研究内容。文中分析了面向对象的互操作技术的产生原因及其在信息技术领域内的重要性,并详细论述了CORBA2.0与互操作有关的内容,包括CORBA的体系结构、界面定义语言、动态调用、对象适配器和ORB之间的互操作。

关 键 词 互操作; 开放系统; 分布式计算; 对象模型; 对象请求代理; 对象适配器
中图分类号 TP311

由于基于大型机、覆盖全企业范围的系统必须与微机以及各部门的服务器相连,分布式计算已经成为信息技术领域中应认真解决的主要问题之一。随着面向对象技术的产生、发展和成熟,它为上述问题的解决找到了合适的途径,并使分布式计算的设计更方便和应用更广泛。分布式计算与面向对象技术的相互推动和促进,导致出现了分布式对象(也称互操作对象)技术。

在分布式对象技术中,应解决的主要问题是,突破由编程语言、进程地址空间和网络界面强加给系统的各种界限,即实现异质环境中对象之间的互操作,同时也是开放系统中的一个研究热点。文献[1]在调查了大量公司和最终用户的基础上得出一个结论:对于计算机领域,有关操作系统、计算机语言以及应用框架的竞争已经结束,现在战场已经开始向互操作对象转移。以上的迹象表明,面向对象的互操作技术是今后几年中信息技术领域内的一项重要研究内容。

在激烈的竞争中,OMG(Object Management Group)的CORBA(Common Object Request Broker Architecture)倍受注目。OMG最初是由3COM、Hewlett-Packard、Novell USG和SunSoft等8个大公司于1989年3月联合成立的集团。其主要目标是开发一种通用方法,使处于异质环境中的分布式对象之间能够实现互操作。现在,OMG已经发展为包括700多个成员在内的国际性集团,其成员几乎遍及了整个计算机工业中的著名厂商。从集团成立之时到现在,OMG推出了两个CORBA版本。特别引人注目的是,1995年6月推出的CORBA2.0(1996年7月,OMG对其作了修订[2]),它把分布式对象之间的互操作技术推上了一个新台阶。

本文对CORBA2.0进行了深入的分析,并对其中与面向对象的互操作技术有关的内容进行了详细的论述,包括:CORBA的体系结构、IDL(Interface Definition Language)语言、动态调用、对象适配器和ORB之间的互操作。

## 1 CORBA的体系结构

CORBA把不同的对象系统集成在一起,使之可以互操作,其基本思想可以用图1表示。

客户是向服务器对象请求服务的实体,服务器对象根据客户的请求,选择相应的操作,完成客户所请求的服务。服务器对象包含界面和对象实现两部分。界面用于描述服务器对象所能提供的

* 1997年9月8日收稿,1997年10月14日修改定稿
* 电子部预研基金资助项目
** 男 26岁 博士生

操作和这些操作所使用的参数,对象实现是具体完成这些操作的代码和数据。客户只能看到服务器对象的界面,而不关心对象实现所处的环境。

CORBA 中的对象由对象引用来识别。客户方和服务器方的对象引用表示法可以不同,通过语言映射,将两者和 ORB 的内部表示法联系起来。这样一来,客户就可以使用自己的表示方法来引用目标对象。

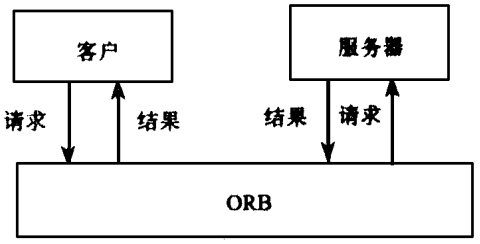


图 1　CORBA 的体系结构

ORB 的主要作用是为客户发出的请求寻找相应的对象实现,并传递请求和对象实现的执行结果。ORB 是这个结构的重要组成部分,对于客户与服务器对象之间的互操作起着关键性的作用,其内部结构如图 2 所示。

在 ORB 的结构中,把用户可见的界面分为三个部分:客户方界面、实现方界面和依赖于 ORB 核心的界面。客户方界面主要包含 IDL 存根、动态调用;实现方界面主要包含静态 IDL 框架、动态框架和对象适配器;客户和服务器对象可以共享 ORB 核心所提供的服务。发出请求时,客户即可以使用 IDL 存根(依赖于目标对象界面的特定存根),也可以使用动态调用界面(不依赖目标对界面的界面)。同时客户也可以直接调用 ORB 提供的某些功能。对象实现通过静态 IDL 框架或动态框架接收客户的请求。当处理客户的请求时,对象实现可以调用对象适配器和 ORB 提供的功能。界面是 CORBA 中的一个重要概念,是 CORBA 解决互操作问题的基础。

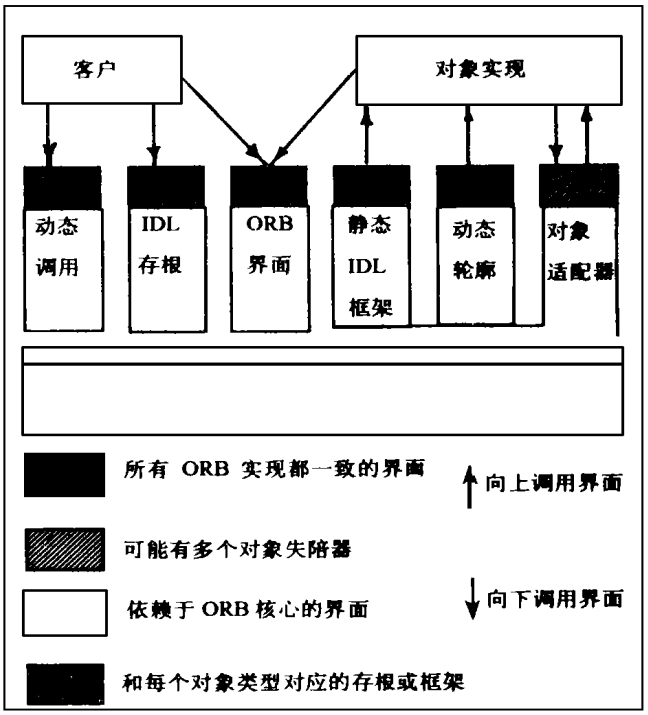


图 2　ORB 的结构

## 2　IDL 语言

在 CORBA 中,IDL 语言用于描述互操作对象的模型,它通过说明对象的界面来定义对象的类型。客户根据 IDL 界面确定相应的对象实现所能完成的操作以及调用它们的方法。引入 IDL 的主要目的是,使界面定义与具体的实现语言无关。IDL 编译器把 IDL 文法映射到客户和对象实现所用的语言,通过对 IDL 界面的编译,分别为客户和对象实现生成与界面有关的组成部分,即客户方的 IDL 存根和实现方的 IDL 框架。客户通过调用 IDL 存根中的例程来使用对象实现所提供的操作;对象实现根据框架编写相应的例程(用于完成在界面中所描述的操作),ORB 通过框架调用这些例程。

在 CORBA 中,客户和对象实现所用的语言不必相同。实际上,实现异种语言的互操作是 CORBA 的主要目标之一。

OMG IDL遵守与C++相同的句法规则,并完全支持C++中的预处理,同时它加入了支持分布式概念的关键字。另外,在OMG IDL中,界面说明没有区分公共和私有部分。

## 3 动态调用

利用IDL存根和静态IDL框架可以实现客户和服务器之间的互操作,但这种方法存在一个明显的缺点:在客户所调用的存根例程组中,每个例程都针对着某个特定对象的某一特定操作,因此要对某对象实现一个操作,客户要调用针对此目标对象的相应的存根例程,也即必须在编译时明确地指定目标服务器。为解决这个问题,CORBA提供了动态调用。动态调用是在执行时说明请求的一种机制。在这种机制中,客户只需(通过一个或者一系列的调用)说明所要调用的对象、将要执行的操作和执行这个操作所需的参数,而不必调用针对某个对象的某一操作的存根例程。动态调用是用于解决ORB之间的互操作问题的一个重要机制。

为增强动态调用的功能,CORBA还引入了界面仓库和实现仓库这两个概念。界面仓库是ORB的一个成分,其中存放着一些永久性界面的定义,它们描述执行时才能确定的与服务器界面有关的信息。利用界面仓库,客户可以确定在编译时无法确定的对象并可以查询它的界面内容,从而发出相应的请求。实现仓库中的信息使ORB定位和激活对象的实现,以此来完成动态请求的任务。

## 4 对象适配器

对象适配器是为对象实现访问ORB服务而设计的。通过对象适配器,ORB提供的服务包括:对象引用的产生和解释、方法的调用、交互的安全保护、对象的激活与终止、对象引用的映射以及对象实现的登记。

CORBA的一个主要目标是把不同的对象系统组合在一起,使之可以互操作。由于不同对象系统中的对象实现都有自己的特点,ORB核心不可能为所有的对象实现提供一个单一的有效界面,但是,通过对象适配器,ORB可以为不同的对象实现组(对ORB界面有相似要求的一组对象实现)分别提供相应的界面。由此可知,对象适配器的引入大大减小了对象实现对ORB的依赖性。

## 5 ORB之间的互操作

由前面所述可知,ORB提供了异质环境中应用对象之间的互操作机制,但是由于不同软件开发商实现的ORB具有各自的特点,所以,如果没有另外的机制,不同ORB上的客户与服务器对象之间无法进行互操作。这是CORBA1.0的一个主要缺点,也是CORBA1.0不能被广泛接受的原因之一。为解决这一问题,首先应该使不同的ORB实现之间能够互操作。在CORBA2.0中,与这一主题相关的内容主要包括以下三个方面:ORB的互操作体系结构,ORB之间的桥接支持,GIOP(General Inter-ORB Protocol)和IIOP(Internet Inter-ORB Protocol)。

### 5.1 ORB的互操作体系结构

ORB的互操作体系结构定义了一个用于描述互操作机制的概念框架。为了能够使独立开发的ORB进行互操作,它还规定了一些必要的约定。

在互操作体系结构中,域是一个非常重要的概念,它表示一个特殊的范围,在这个范围中的实体具有共同的特点,并遵守相同的规则。ORB之间的互操作就是使客户的请求信息和服务器的返回结果能够透明地通过不同域之间的界限。完成这一任务的基本思想是利用桥机制,根据源域和

目标域中的有关规定将通过域界限的数据进行翻译。

如图 3 所示, CORBA 中的桥接方法分为两种: 直接桥和间接桥。直接桥把与互操作有关的元素从源域的内部形式直接转换为目标域的内部形式。间接桥由两个"半桥"组成, 每个半桥和一个域相连, 它把与互操作有关的元素从源域的形式转换为公共形式, 或者把元素的公共形式转换为目标域的内部形式。

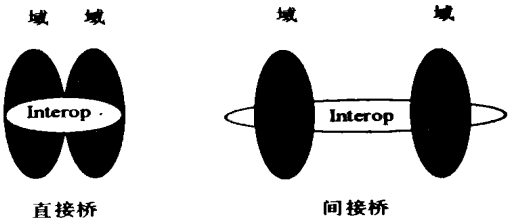


图 3　ORB 之间的桥接方法

### 5.2　ORB 之间的桥接支持

ORB 之间的桥接支持的主要目标是提供足够的 ORB API 并说明一些必要的约定, 使用户可以很方便地构造 ORB 之间的互操作桥。另外, 还可以利用这些 API 来完成 ORB 和非 ORB 系统(如 Microsoft 的公共对象模型 COM )之间的互操作。

根据桥单元所在位置的不同, 可以把实现桥的方法分为两种, ORB 层桥和请求层桥。ORB 层桥对域内容进行翻译的代码(即桥单元)在 ORB 的内部。它是利用 ORB 的内部 API 开发的, 支持这种桥的 ORB 必须提供额外的 ORB 服务以及存根和框架代码。请求层桥的桥单元在 ORB 的外部。它是利 ORB 的公共 API 开发的。其基本思想如下: 客户方 ORB 使用 DSI(Dynamic Skeleton Interface)将客户的请求交给桥中的一个代理对象, 代理对象将请求内容转化为服务方 ORB 能够理解的格式, 然后代理对象通过 DII(Dynamic Invocation Interface)向目标服务器调用请求的操作。

### 5.3　GIOP 和 IIOP 协议

GIOP 说明 ORB 之间通信的文法和信息格式。它不使用高层的 RPC 机制, 并对低层支撑软件的依赖性较弱。对任何面向连接的传输协议做极少量的假设后, 其上都可直接运行 GIOP。

要实现 ORB 之间的通信, 只有 GIOP 是不够的。只有当 GIOP 的信息映射到网络的传输层以后, 才能完成这个任务。IIOP 把 GIOP 信息映射到 TCP/P 协议上, 即说明如何在 TCP/IP 连接上交换 GIOP 信息。引入 IIOP 后, 异种 ORB 上的对象之间的相互访问过程如图 4 所示。

ESIOP 是 GIOP 信息向其他网络协议(如 DCE 的 RPC)的映射, 不同的 ORB 实现可以直接在 IIOP 上运行, 也可以在某个 ESIOP 上运行。但是, 对于后者, ORB 实现必须与一个"半桥"相连。"半桥"的作用是使与之相连的 ORB 实现能够与基于 IIOP 的 ORB 实现互操作。

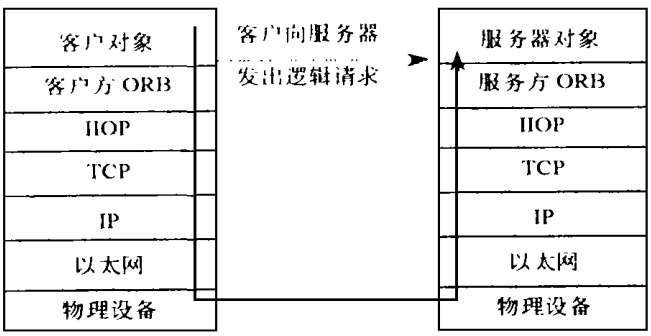


图 4　异种 ORB 上的对象之间的访问过程

## 6　结 束 语

本文根据 CORBA 2.0 的基本思想, 对面向对象的互操作技术进行了详细论述, 这些内容对分布式软件开发者和信息技术人员的实际工作以及认识未来几年中主流技术的发展都有重要意义。

## 参 考 文 献

1 Mark Betz. Inter operable objects. Dr Dobb's Journal, 1994:18~39
2 OMG, X/Open. The common object request broker: architecture and specification. Revision 2.0, Updated July, 1996
3 Valdes Ray. Introducing interoperable objects. Dr Dobb's Journal, 1994/1995:4~6
4 Minton Gabriel. Programming with CORBA. Unix Review, 1996:29~44
5 Orfali Robert, Dan Harkey. Client/server with distributed objects. BYTE, 1995:151~160
6 Valdes Ray. Implementing interoperable objects. Dr Dobb's Journal, 1994/1995:62~66
7 Minton Cabriel. IIOP specification: a closer look. Unix Review, 1997:41~48
8 W. Rosenberg and Denney. Understanding DCE. Open System Foundation, 1992
9 苏 森, 唐雪飞. 开放系统中的互操作性. 计算机应用, 1997, 17(6):4~7

# Object-oriented Interoperability Technology

Su Sen Tan Xuefei Liu Jinde
(College of Computer Science and Eng., UEST of China Chengdu 610054)

**Abstract** In IT area, object-oriented interoperability technology is going to be mainstream over next few years. The reason that the objcet-oriented interoperability technology appeared and its usefulness in IT area are analyzed. The main components that provide interoperability in CORBA 2.0 are clearly described. Those are CORBA architecture, IDL language, dynamic invocation, object adapter and the interoperability between ORB.

**Key words** interoperability; open system; distributed computing; object model; object request broker; object adapter

编辑 黄 辛