ISSN1000-9825 Journal of Software 软件学报 2000,11(4):540~544

基于多界面的 COM/CORBA 互操作*

兑继英 汪芸 顾冠群

(东南大学计算机科学与工程系 南京 210096)
(计算机网络和数据集成国家教育部重点实验室 南京 210096)
E-mail: jdui@seu.edu.cn

摘要 在 OMG(object management group)的 COM(common object model)/CORBA(common object request broker architecture)互操作规范中,COM/CORBA 互操作采用桥的方式实现.尽管这种方法适用范围较广,但其性能较差,无法实现透明的对象生命周期映射,而且没有解决 COM 和 CORBA 应用程序的移植问题.该文提出一种基于多界面对象集成方法的 COM/CORBA 互操作模型,使客户可以直接访问异种对象,从而提高互操作的性能,增强互操作的透明性,还可以实现应用程序在异构环境下的移植.

关键词 COM(common object model),CORBA(common object request broker architecture),COM/CORBA 互操作,多界面.

中图法分类号 TP311

COM(common object model)和 CORBA(common object request broker architecture)应用的增长使得 COM/CORBA 互操作的需求不断上升,因此,很多 CORBA 产品都遵照 OMG 组织(object management group)制订的 COM/CORBA 互操作规范实现了 COM/CORBA 互操作.规范中的互操作方法通过桥来连接两个异构系统[1],桥负责将客户请求转换为对异种对象的请求,并将这个请求发送给异种对象,最后对请求结果进行转换并返回给客户.

显然,与同种对象模型中客户对服务器的调用相比,这种方法在性能方面下降了很多.另外,在 COM 和 CORBA 对象生命周期机制中存在的差异使得 COM 和 CORBA 的对象生存周期无法透明地进行映射.例如,COM 客户能够控制 COM 对象的生命周期,但在互操作的情况下,COM 客户无法控制 CORBA 对象的生命周期.此外,规范中也没有解决 COM 和 CORBA 应用的移植问题.因此,文中提出使用多界面对象集成方法来实现 COM/CORBA 互操作,在提高性能和增强透明性的同时,还可以实现应用程序在异构环境下的移植.

1 多界面对象集成方法

多界面对象集成方法 MIOIM(multi-interface object integration method,简称 MIOIM)的基本思想是将对象实现为同时具备 COM 和 CORBA 界面的多界面对象,使得服务器既可以支持 COM 客户,也可以支持 CORBA 客户.另一方面,通过为应用程序建立统一代理,客户可以访问各种对象模型中的对象,从而实现 COM 和 CORBA 的集成.

1.1 基本模型

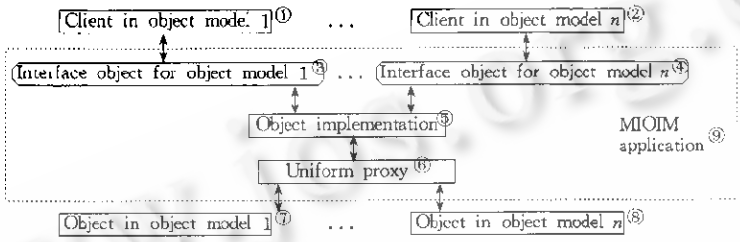
不论是 COM 还是 CORBA,对象都通过界面来展示它的功能,客户通过对象的界面来访问对象[1,2].对应用

* 本文研究得到国家 863 高科技项目基金(No.863-511-9704-002)和教育部应用基础项目基金(No.98046)资助.作者兑继英,1971 年生,博士生,主要研究领域为计算机网络、电子邮件,分布式处理.汪芸,女,1967 年生,博士,副教授,主要研究领域为集成框架,分布式处理.顾冠群,1940 年生,教授,博士生导师,中国工程院院士,主要研究领域为协议工程,高性能网络,集成框架,CIMS 网络,EDI.

本文通讯联系人:兑继英,南京 210096,东南大学计算机科学与工程系

本文 1998-11-25 收到原稿,1999-05-24 收到修改稿

来说,不论它的对象是通过COM界面展示,还是通过CORBA界面展示,其对象实现都是相同的.MIOIM的思想是将对象设计和实现为具有多种界面的对象,以满足不同客户的需求.同时,客户也可以通过统一代理的方式实现对各种对象的访问.MIOIM的基本模型如图1所示.其中,对象模型$i(i=1,2,\ldots,n)$代表不同的对象模型.对象模型i中的界面对象是为对象模型i中的客户开发的界面对象,负责将客户的请求转换为对对象实现的调用,并将调用结果按照它所支持的界面格式返回给客户.对象实现是指独立于具体对象模型的对象功能实体.统一代理为客户访问同种或异种对象提供了一个统一的界面,它负责按照目标对象所在的对象模型的访问方式访问对象,当对象实现作为客户时,可以通过统一代理来访问各种模型中的对象.使用统一代理访问对象的客户程序称为多界面客户,采用MIOIM实现对象的应用程序称为多界面服务器,多界面客户和多界面服务器统称为多界面应用程序.



①对象模型1中的客户,②对象模型n中的客户,③对象模型1中的界面对象,④对象模型n中的界面对象,⑤对象实现,⑥统代理,⑦对象模型1中的对象,⑧对象模型n的对象,⑨多界面应用程序.

Fig.1 Model of MIOIM
图1 MIOIM的基本模型

与规范中的互操作机制相比,MIOIM具有以下几个优点:

(1) MIOIM中的界面对象、对象实现和统一代理实现在同一个程序中,由于分布对象计算系统的性能较低[3],其访问代价比进程内的直接访问高得多,因此,MIOIM在性能上比互操作有很大的提高;

(2) 在MIOIM中,针对不同对象模型的客户开发的界面对象可以具有不同的生命周期,因此,不需要对生命周期进行映射;

(3) 由于使用了界面对象和统一代理,对象实现完全独立于所使用的对象模型,因此,可以通过条件编译的方法,根据应用所在的环境定制界面对象,从而实现应用程序在不同对象系统下的移植;

(4) 可以针对其他对象模型来增加界面对象和扩充统一代理,因此,除了支持COM/CORBA互操作外,还可以方便地支持其他对象模型,例如,OODCE和JAVA RMI.

1.2 实现方法

MIOIM的实现方法有两种.一种方法是将统一代理和界面对象集成到COM或CORBA的桩和构架中.这种方法可以保留已有的开发界面,用户可以按照原来的方式开发应用程序,因此具有较高的透明性,有利于已有应用的迁移,但实现代价较高.另一种方法是定义新的桩和构架,基于已有的桩和构架实现界面对象和统一代理.由于定义了新的开发界面,这种方法的透明性不如前者,不利于已有应用的迁移,但实现代价较低.

除此之外,COM和CORBA还支持动态方式实现客户和服务器.在这种方式下,用户无需使用桩和构架,而通过系统调用可直接生成请求或对请求进行处理.MIOIM可以改造或重新定义相关的系统调用,对于客户来讲,可以根据它所访问的对象类型,对其动态生成的请求进行转换并发送;对于服务器来说,则可以将请求转换之后再交给服务器.

MIOIM可以沿用互操作规范中的界面映射和界面组成映射规则,这样,一方面避免了重复工作,另一方面也有利于客户程序在两种互操作系统上的移植.由于多个界面对象使用同一个对象实现,MIOIM需要解决界面对象对对象实现的共享问题.COM和CORBA有各自的对象实现共享机制,可以在此基础上定义和实现对象实现的共享机制及并发控制机制.

2 MIOIM 的适用范围

由于界面对象、对象实现和统一代理实现在同一个程序中,使得 MIOIM 需要在 COM 和 CORBA 的混合环境下实现,因而限制了它的应用范围.下面对 MIOIM 和规范中的互操作方法的实现环境进行比较.

分布对象计算系统所支持的环境包括 3 个要素:操作系统、编程语言和网络环境.设 COM 所能支持的操作系统集合为 S_{com},编程语言集合为 L_{com},网络环境集合为 N_{com};相应的 CORBA 所能支持的环境表示为 S_{corba},L_{corba}和 N_{corba};设某个操作系统 s_i所能支持的编程语言集合为 L_i,网络环境集合为 N_i.假设$\exists s_i, s_i \in S_{com} \wedge s_i \in S_{corba} \wedge L_i \cap L_{com} \cap L_{corba} \neq \varnothing \wedge N_i \cap N_{com} \cap N_{corba} \neq \varnothing$.

对于规范中的互操作方法,COM 客户访问 CORBA 对象的互操作应用环境为

$$E_{com,corba} = E_{com} \times E_{corba} = \left(\bigcup_{s_i \in S_{com}} s_i \times (L_i \cap L_{com}) \times (N_i \cap N_{com})\right) \times \left(\bigcup_{s_i \in S_{corba}} s_i \times (L_i \cap L_{corba}) \times (N_i \cap N_{corba})\right).$$

对于 MIOIM,COM 客户可以通过两种方式访问 CORBA 对象.一种是通过为 COM 客户开发统一代理,另一种是将 CORBA 对象实现为多界面对象.设这两种方法所支持的应用环境分别为 E_1 和 E_2,则

$$E_1 = \left(\bigcup_{s_i \in S_{com} \cap S_{corba}} s_i \times (L_i \cap L_{com} \cap L_{corba}) \times (N_i \cap N_{com} \cap N_{corba})\right) \times \left(\bigcup_{s_i \in S_{corba}} s_i \times (L_i \cap L_{corba}) \times (N_i \cap N_{corba})\right),$$

$$E_2 = \left(\bigcup_{s_i \in S_{com}} s_i \times (L_i \cap L_{com}) \times (N_i \cap N_{com})\right) \times \left(\bigcup_{s_i \in S_{com} \cap S_{corba}} s_i \times (L_i \cap L_{com} \cap L_{corba}) \times (N_i \cap N_{com} \cap N_{corba})\right).$$

$\because E_1 \subseteq E_{com,corba}, E_2 \subseteq E_{com,corba}$,

$\therefore E_1 \cup E_2 \subseteq E_{com,corba}$.

由此可以看出,MIOIM 的适用范围较小.具体地,令

$$A = \bigcup_{s_i \in S_{com} - S_{corba}} s_i \times (L_i \cap L_{com}) \times (N_i \cap N_{com}), \quad B = \bigcup_{s_i \in S_{com} \cap S_{corba}} s_i \times ((L_i \cap L_{com}) \times (N_i \cap N_{com}) - (L_{corba} \times N_{corba})),$$

$$C = \bigcup_{s_i \in S_{corba} - S_{com}} s_i \times (L_i \cap L_{corba}) \times (N_i \cap N_{corba}), \quad D = \bigcup_{s_i \in S_{com} \cap S_{corba}} s_i \times ((L_i \cap L_{corba}) \times (N_i \cap N_{corba}) - (L_{com} \times N_{com})),$$

则 $E_{com,corba} - E_1 \cup E_2 = (A \cup B) \times (C \cup D)$.

当客户方为 CORBA、服务器方为 COM 时,有类似的结果.令 MIOIM 方法中为 CORBA 客户并发统一代理的应用环境集合为 E'_1,将 CORBA 对象实现为多界面对象的应用环境集合为 E'_2,则

$$E_{corba,com} - E'_1 \cup E'_2 = (C \cup D) \times (A \cup B).$$

上述结果说明,在某些情况下不能使用 MIOIM,例如,COM 客户使用 Visual Basic 开发,CORBA 对象实现在 UNIX 平台上.目前 DCOM 正在向其他操作系统移植,CORBA 也在不断扩充所支持的编程语言.随着 COM,CORBA 应用环境的相互渗透,今后 MIOIM 的适用范围也会不断扩大.在理想情况下,$S_{com} = S_{corba}, L_{com} = L_{corba}, N_{com} = N_{corba}$,则 $E_{com,corba} - E_1 \cup E_2 = E_{corba,com} - E'_1 \cup E'_2 = \varnothing$,MIOIM 和规范中的互操作方法的适用范围相同.

3 实例介绍

以下是一个简单的实验,内容为将一个名为 grid 的 CORBA 对象改造为同时具备 CORBA 界面和 COM 界面的对象,初步验证了 MIOIM 的可行性和性能.grid 的 OMG IDL 和 MIDL 界面定义如下:

```
//OMG IDL definition of grid
interface grid {
  readonly attribute short height; //height of the grid
  readonly attribute short width; //width of the grid
  void set (in short n, in short m, in long value); //set the element [n,m] of the grid;
  long get (in short n, in short m); //return element [n,m] of the grid;
};
//MIDL definition of grid
[...]
```

```
interface IGrid: IUnknown {
  import "unknwn.idl";
  HRESULT get_height([out] short *retval); //height of the grid
  HRESULT get_width([out] short *retval); //width of the grid
  HRESULT set([in] short n, [in] short m, [in] long value); //set the element [n,m] of the grid;
  HRESULT get([in] short n, [in] short m, [out] long *retval); //return element [n,m] of the grid;
};
```

此外,实验中还实现了grid的COM服务器,并按照规范中的互操作机制为grid定制了COM代理和CORBA代理.实验使用同一个客户程序对3种不同的访问方式进行测试,测试分本地和远程两种,测试内容为调用每个操作所花费的平均时间(单位:μs).由于4个操作的测试结果差别很小,这里只列出get操作的测试结果,见表1和表2.

Table 1 Performance test based on COM client

表1 基于COM客户的性能测试

	Local①	Remote②
MIOIM	0.126 8	1.833 0
COM proxy③	4.635 8	7.847 5
COM object④	0.125 2	1.827 7

①本地,②远程,③COM代理,④COM对象.

Table 2 Performance test based on CORBA client

表2 基于CORBA客户的性能测试

	Local①	Remote②
MIOIM	4.320 5	8.335 6
CORBA proxy③	4.662 6	8.792 5
CORBA object④	4.294 7	8.274 0

①本地,②远程,③CORBA代理,④CORBA对象.

从测试结果可以看出,MIOIM在性能上有显著提高,与客户直接访问对象的性能基本相同.从测试结果还可以看出,不论本地调用还是远程调用,COM的性能都高于CORBA,因此,MIOIM对COM代理性能的提高更为显著.

4 结束语

本文提出使用多界面对象集成方法来实现COM和CORBA的互操作.该方法的主要优点在于提高了互操作的性能,此外,还提高了透明性、实现了可移植性.由于目前MIOIM在适用范围上还受到限制,因此,还不能完全代替规范中的互操作方法,而可以作为规范中的互操作方法的扩展来使用.

参考文献

1 Wang Yun. CORBA Technique and Its Applications. Nanjing: Southeast University Press, 1999
(汪芸. CORBA技术及其应用. 南京:东南大学出版社,1999)

2 Brockschmidt K. Inside Ole 2. Redmond: Microsoft Press, 1998

3 Schmidt D C, Gokhale A S, Harrison T *et al*. High-Performance and system architecture for real-time CORBA. IEEE Communications Magazine, 1997,35(2):72~77

COM/CORBA Interworking Based on Multi-Interface

DUI Ji-ying WANG Yun GU Guan-qun

(Department of Computer Science and Engineering Southeast University Nanjing 210096)

(Key Laboratory of Computer Network and Data Integration Ministry of Education Nanjing 210096)

Abstract In the OMG's COM/CORBA interworking specification, interworking between COM and CORBA is implemented via bridges. Though the interworking model is flexible, it needs heavy overheads which leads to poor performance. Moreover, there are problems in its object life-cycle mapping. It does not provice portability

either. In this paper, a COM/CORBA interworking model is proposed based on multi-interface object integration method. In the new model, a client can directly access a foreign object, therefore interworking performance is improved remarkably. In addition, it enhances interworking transparency and provides application portability between COM and CORBA environments.

Key words COM(common object model), CORBA(common object request broker architecture), COM/CORBA interworking, multi-interface.