异构环境下用于构件管理的CORBA接口的设计与实现

The Design and Implementation of the CORBA Interface Used for Managing the Component in the Heterogeneous Environment

赵 慧 刘西洋 陈 平 蔡希尧
(西安电子科技大学软件工程研究所 西安 710071)

Abstract This paper discribes the design and the implementation about the CORBA2 interface for the ENVY/Developer. ENVY/Developer is a collaborative component development systems for Smalltalk developers. It makes enterprise-wide reusable component repositories a reality. This interface can be used in heterogeneous environment.

Keywords CORBA, ENVY/Developer, Component, Heterogeneous environment

1 引言

大型分布式系统往往由多种平台组成,由此引起的异构特性加剧了网络上应用开发和维护的困难。OMG(Object Management Group)制定的CORBA(Common Object Request Broker Architecture)规范为分布式系统提供了一个面向对象的计算模型,基于CORBA标准的应用屏蔽了平台、语言以及厂商的信息,依赖ORB(Object Request Broker)提供的服务,使得对象在异构环境下透明地通信。

我们在进行基于构件的软件重用的研究中,围绕构件设计了构件管理工具、构件开发工具和构件存储库,整个系统架构在分布网络环境下。图1表示了系统结构图。

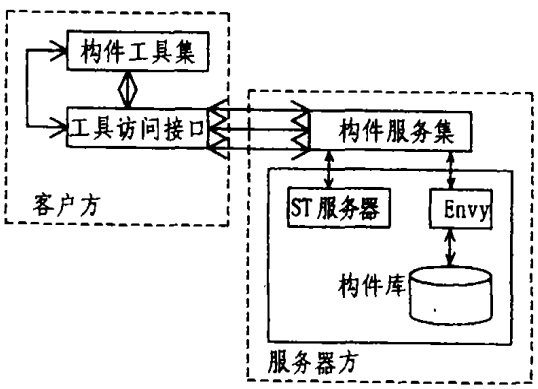

图1 系统结构

服务器方由多种平台构成,采用为Smalltalk开发人员提供的协作的部件开发系统ENVY/Developer存储构件。ENVY/Developer支持并行开发、共享库、版本控制、配置管理和分布式开发。构件用C++语言实现。客户方大量与构件有关的工具的实现不拘泥一种开发平台。由此,系统出现异构环境互操作的问题。

RPC(远程过程调用)是网络上的高层协议,允许网络上的应用通过特定的网络上的过程调用来实现,从而隐蔽了下层网络的细节。RPC实现的是客户方和服务器方之间通信的逻辑系统,用于支持网络上的应用开发。利用RPC,客户方可以发送过程调用的请求给服务器方,服务器方在接收到这些请求后调用本地过程执行客户方请求的操作,并将执行的结果返回给客户方。

DCE通过基础的分布式设施和数据共享设施在网络上提供增值服务,为应用开发人员提供具有良好的互操作性的开发环境。开发人员通过DCE可以广泛地使用所有的系统和设施而无需考虑用户、应用程序和其它所需资源的位置,从而使得用户可以更有效、更充分地使用网络上的资源。然而,RPC和DCE都存在一些明显的不足之处:

(1)客户程序和服务器程序之间的调用关系是静态的;

(2)调用关系的静态性质导致了所构建的系统也是静态的,即一旦有程序发生变化(尤其是服务器方程序),那么整个系统必须重新编译生成,原有的系统就会被抛;

(3)客户程序和服务器程序必须用RPC或

DCE的实现语言来编制，因为这些程序需要同RPC或DCE提供的库函数或接口界面捆绑在一起。这样就限制了用户的可选语言范围。

因此，我们采用IONA公司的Orbix ORB为ENVY/Developer建立CORBA2接口。

2 CORBA及其运行环境

OMG的中心任务是在分布式环境下基于对象技术，建立一个体系结构和一组规范，实现应用的集成，使得基于对象的软件部件在分布异构环境中可重用、可移植和可互操作。OMG提出的对象管理体系结构OMA(Object Management Architecture)参考模型由以下五个部分组成：

(1)对象请求代理ORB：使对象在分布式环境中透明地收发请求和响应，它是构造分布式对象应用，使应用在不同层次的异构环境下互操作的基础。目前流行的ORB产品有IONA公司的Orbix系列、HP公司的ORB Plus。

(2)对象服务(Object Services)：是为使用和实现对象而提供的基本服务集。对于分布式应用，这些是基本服务，它们独立于应用领域，例如生命周期服务定义了对象创建、删除、拷贝和移动的方式，它不考虑对象在应用中如何实现。

(3)公共设施(Common Facilities)：是为多个应用提供的共享服务集合。

(4)应用接口(Application Interfaces)：相对应于传统的应用表示，可由单个厂商提供，是参考模型中的最高层，OMG对此没有作规定。

(5)域接口(Domain Interfaces)：与对象服务和公共设施的作用类似，但是它面向特定的应用领域。

CORBA规范细化了OMG的ORB部件的特征和接口，CORBA2.0的主要特征有：相当于分布对象管理器的ORB核心、OMG接口定义语言(IDL)、接口库(Interface Repository)、语言映射、存根(Stub)和框架(Skeletons)、静态调用、动态调用和分发、对象适配器以及ORB间协议(IIOP)。OMG已于1998年9月推出了CORBA3.0的技术规范，增加了新的特性：

(1)将直观应用程序工具和环境用于CORBA组件的设计；

(2)支持CORBA对象在异步实时消息队列中进行传输；

(3)采用QoS技术规范；

(4)增加了新的技术规范，以便实现与传统环境的结合。

CORBA既允许在本地创建对象实例也允许远程创建对象实例。若实例是远程的，则可以从任意符合CORBA标准(CORBA－Compliant)的客户访问这个实例；若是本地实例，只能由创建它的应用访问。如图2所示，在创建CORBA实例的每一台主机上，有一个Daemon称作CORBA服务器，负责处理对象的创建/删除以及客户方发送的请求，为处理这些请求，服务器访问两个数据库：接口库和实现库，接口库是表示接口定义元素的持久类型库，它的创建与维护基于IDL源文件提供的信息；实现库是包含CORBA对象定义的实现的数据库。无论客户方应用何时创建一个远程对象，服务器使用实现库来访问实现请求的CORBA对象代码，并且运行初始化CORBA对象的一个应用。客户方由一个代理指针获取对象的创建，这个指针位于客户方的地址空间，指向服务器应用创建的真正的实例。由此，不允许客户方直接控制实例，而只能透明地通过代理实例访问。

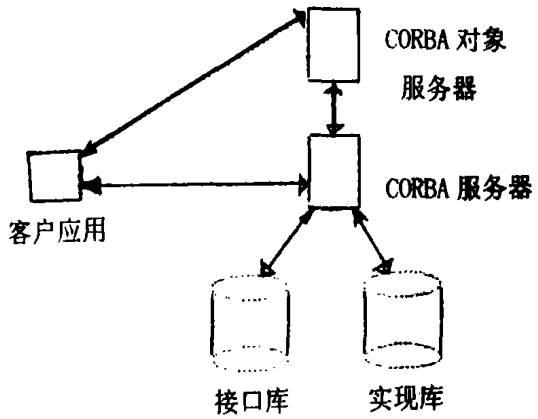

图2 CORBA运行应用环境。

3 设计实现

3.1 总体框架

从图1分析得到，我们需为Envy服务器设计CORBA接口，总体框架结构如图3表示。客户方的

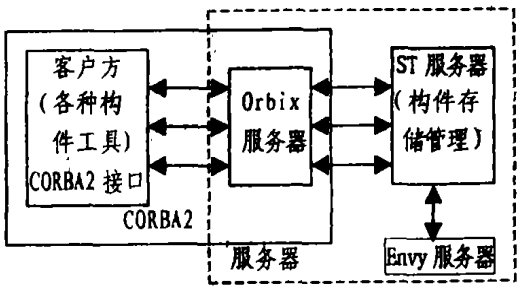

图3 Envy服务器CORBA2接口总体框架

各种构件工具通过CORBA2接口向Orbix服务器和ST服务器发出请求。构件存储由Envy/Develop-

er 管理，前端构件工具通过 CORBA 接口访问 Orbix 服务器。

3.2 技术途经

根据图 3 的分析，服务器方的所有服务器进程装载于一台计算机上，Orbix 服务进程接收多个客户进程的请求，然后根据请求分发到 ST 服务进程，ST 服务进程接收到请求后解析请求，并访问 Envy 服务器，在 Envy/Developer 中存取构件。基于这样的过程，我们采用的技术途经是：CORBA2 部分基于 Orbix 来实现，通过 Orbix 提供的 ThreadFilter 机制分发客户进程的请求，每个客户方的请求指派服务器进程的一个线程处理。Orbix 服务进程与 ST 服务进程间的通讯采用共享内存的方式完成。由于 Smalltalk 的进程机制是在 Smalltalk 虚拟机上实现的，Smalltalk 实现的进程与 C++实现的进程直接通讯时，解决同步互斥问题比较困难，因此我们利用 C++实现的动态链接库作为二者的桥，实现同步和互斥，而 Smalltalk 进程只需通过虚拟机 API 与动态链接库进行数据交换。

3.3 实现技术

实现这个接口的关键与难点在于：

(1)IDL 文件中对象接口的定义。我们围绕构件库，抽取有关构件的最基本的操作，如存取构件的数据模型、查询构件的存在等作为对象的基本方法，为构件工具提供基本的操作对象。

(2)对象的实现。针对构件系统，包括为客户创建一个新的对象，分发客户的请求等，这部分的实现充分利用 Orbix 提供的 ThreadFilter 机制。ThreadFilter 是 Orbix Filter 类的子类，在 Orbix 中有两种形式的 Filter：per－process 和 per－object，per－process filter 可以观察所有进入及离开客户或者服务器进程地址空间的操作和属性调用，inRequestPreMarshal()函数能够创建一个线程来处理一进入的请求，即每个请求一个线程模式；per－object filter 则适用于单独的对象。根据设计，我们采用每个请求一个线程模式以及 ThreadFilter 的子类得出的线程模型之一：每个客户一个线程(Thread per Client)来实现服务器进程对客户请求的分发。

(3)对象的实现通过动态链接库与 Smalltalk 进程通讯。一是 Smalltalk 进程作为主控进程接收 Orbix 服务器的请求，通过 PlatformFunction (Smalltalk 提供的与 C 语言接口的类)的实例方法 coroutineCallThreadKey 建立多个线程，并行处理 Orbix 服务器的请求；二是通过 Smalltalk 的 OSObject 类协调 C 的内存访问机制与 Smalltalk 的内存访问机制，转换 CORBA 的数据类型使其与 Smalltalk 的对象保持一致。

(4)并发控制。

· 客户进程对 Orbix 服务器中的对象的访问：由两级并发控制完成，基本的 Orbix ORB 库是安全线程的，保证多个并发线程不会破坏对象内部的变量和表，在应用级采用 Mutex 和 Semaphore 机制；

· Orbix 服务器进程与动态链接库对共享内存的访问：在应用级采用 Mutex 和 Semaphore 机制；

· Envy/Developer 构件库的访问：Envy/Developer 本身提供了部件级的访问控制，保证部件的完整性；

· Smalltalk 进程与动态链接库的同步：在实现级通过 PlatformFunction 类的异步调用实例方法实现。

以上并发控制是接口实现的关键。

结束语 本文叙述了在 Orbix Desktop2.2 上实现的 Envy/Developer 的 CORBA2 接口。完整的构件库是基于构件的软件开发环境的关键，我们选择 Envy/Developer 作为构件的存储库。Envy/Developer 是为 Smalltalk 开发者提供的部件存储库，能够在部件层进行并发访问、对部件进行版本控制和配置管理以及支持网络开发，它的虚拟空间可达到 16GB。CORBA 是 OMG 提出的在分布式环境下，面向对象的软件部件能够可重用、可移植、可互操作的规范。由于构件工具及构件的实现不局限于 Smalltalk 语言，我们设计实现了 Envy/Developer 的 CORBA2 接口，使得构件系统的异构平台能够互操作。

参考文献

1 Jacobson I, Griss M, Jonsson P. Software Reuse Architecture Process and Organization for Business Success. ACM Press, 1997

2 IONA Technologies PLC. Orbix2 Programming Guid. 1997

3 Vinoski S. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. IEEE Communication Magazine, 1997, 2

4 汪 芸，兑继英，顾冠群. 开放分布式处理的新发展——CORBA 规范. 计算机科学，1997，24(3)

5 IONA Technologies PLC. Orbix2 Programming References. 1997

6 IBM VisualAge for Smalltalk Programming Reference. 1997