国 防 科 技 大 学 学 报
第21卷 第2期 JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY Vol.21 No.2 1999

分布式软件平台StarBus中接口池的研究与实现*

刘惠 周明辉 周立

(国防科技大学计算机系 长沙 410073)

摘 要 按照CORBA标准,阐述了接口池在StarBus系统中的作用,介绍了StarBus系统中接口池的设计,说明了接口池的内容与结构,实现了StarBus系统中的接口池服务,为在StarBus系统中实现动态激活提供了保证。

关键词 CORBA标准,StarBus系统,接口池,动态激活

分类号 TP311

The Study and Implementation of Interface Repository in StarBus System

Liu Hui Zhou MingHui Zhou Li

(Department of Computer·NUDT, Changsha, 410073)

Abstract In compliance with the CORBA specification, the paper has elaborated the design of Interface Repository and the functions of the Interface Repository in Star Bus system, illustrated the contents and architecture of Interface Repository, introduced in detail the implementation of Interface Repository Service.

Key words CORBA specification, StarBus system, interface repository, dynamic invoke

基于CORBA的分布式应用中,分布式对象提供的服务被声明成公共接口。为此CORBA标准定义了接口定义语言OMG IDL(Interface Definition Language)来描述服务对象的公共接口。接口定义包括了对象支持的一系列操作,操作中涉及的参数及其类型,操作的属性及执行操作时可能产生的异常等信息。用户可以根据这些接口定义来构造访问服务对象的服务请求。

用户在正确地向服务对象发出服务请求前,必须首先获得服务对象的接口定义。根据CORBA标准的定义,用户可以通过两种途径来访问由IDL语言具体描述的服务对象的接口定义。

(1) 由IDL编译器根据服务对象的IDL接口定义生成相应的存根(stub)进程(所谓存根进程是一个远程对象在本地进程中的表示,实际上是一组可执行操作的静态请求)。用户可以使用这个存根进程来访问服务对象。这种方法在编译时刻服务请求已经构造好,称为静态激活。

(2) 用户在程序编译时只知道被请求对象的对象引用,而在程序运行时刻动态地获得相应的请求操作信息,然后再正确地构造服务对象请求。这种方法被称为动态激活。

1 接口池的引入

接口池是为实现动态激活而设计的,提供了一个存储与接口定义有关信息的场所和对其进行访问的相应机制。系统在接口池中保存了由IDL语言描述的所有服务对象的公共接口定义,包括操作名、操作参数的数据类型,供用户动态地进行查询。

例如一个提供银行存款、取款、查询账目服务的对象,可以通过IDL语言定义公共接口,提供存款(deposit)、取款(takeout)和账目查询(access)三种操作。通过对公共接口的说明还可以进一步获得执行操作所使用的参数,以及操作可能会产生的异常等。

如图1所示,在使用动态激活时,用户首先访问名字服务器,获得服务对象的对象引用;然后通过对象引用在接口池中进行搜索,查询服务对象公共接口的操作信息,动态地构造服务请求。

* 国家863高技术项目资助
1998年7月2日收稿
第一作者:刘惠,女,1971年生,硕士生

2 接口池的设计

CORBA 标准在接口池中使用对象的形式来组织和管理接口定义, 这些对象分为接口对象和类型对象, 其中接口对象表示接口定义和操作定义, 类型对象描述了操作中用到的参数的 IDL 类型信息。

IDL 文件中的接口定义之间存在两种关系, 一种是包含与被包含关系, 另一种是继承关系。接口池对象之间也相应地存在这两种关系, 而且接口池对象的组织结构与它们所表示的接口定义的结构类似。为了完整地表达接口对象之间的相互关系, 并且提供一个可完成所有操作的最小接口池对象集合, 我们在接口池中实现了13个类。这些类被分为两个部分: 抽象类和具体类。其中抽象类只能被其它类继承使用而不能实例化, 并且对于用户而言是透明的。这些抽象类之间相互关系如图2所示。

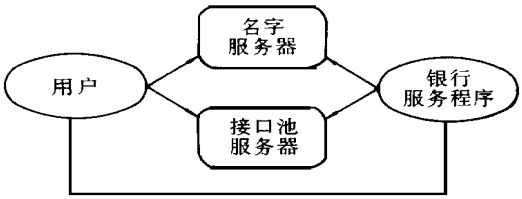

图1 使用接口池的动态激活示例

Fig. 1 demonstration of dynamic invoking using interface repository

IRobject 类是所有保存在接口池中的对象的父类, 描述了接口池中所有其他对象应具备的基本信息。包含类(Container)和被包含类(Contained) 表示接口池对象之间的包含关系, 其中 Contained 抽象类是接口池中所有被其它接口池中的对象所包含的对象的父类, Container 抽象类是接口池中所有包含其它对象的接口池对象的父类, 通常用来形成接口池中对象之间包含关系的层次结构。IdlType 类是所有与表示 IDL 语言类型有关的对象的父类, 而 TypedefDef 类则是某些复杂类型对象的父类。

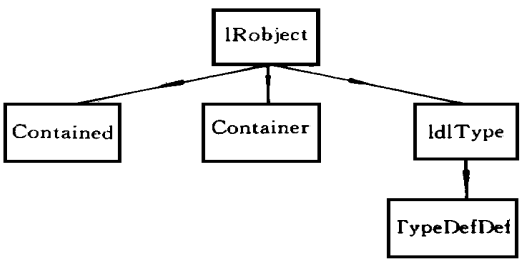

图2 接口池抽象类关系描述

Fig. 2 Abstract classes implemented in interface repository

具体类由抽象类派生而来, 用以定义接口池中保存的服务对象的接口定义。

3 接口池服务的关键实现技术

根据CORBA 标准所定义的分布应用模型, 一个分布对象由对象代理和对象实现两部分组成, 如图3所示。在这个分布应用模型中, 对象实现在服务方, 真正实现服务对象所提供的各项功能; 对象代理位于用户方, 是应用程序与对象实现之间进行通讯的接口机制。

因为要提供在分布异构环境下访问接口池的服务, 所以在实现接口池服务时, 把它设计为一个典型的 CORBA 分布应用的形式。整个接口池的服务程序由以下四个部分组成:

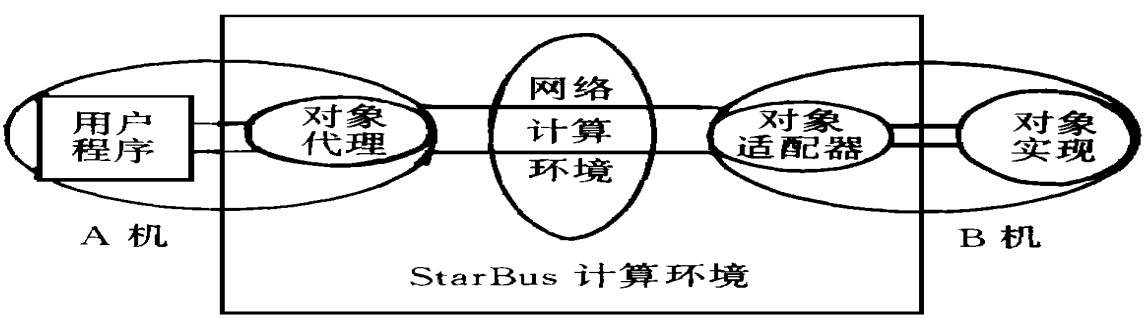

图3 分布式对象结构

Fig. 3 Structure of distributed objects

(1) IDL 语言描述的接口池公共访问接口,用户可以根据这些公共接口来访问接口池;

(2) 用户方的对象代理,负责对应用程序的服务请求进行编码并发出请求;

(3) 服务方的服务程序,将用户的服务请求转发到相应的对象实现;

(4) 服务方的对象实现,真正实现接口池对象服务,以及对接口池中的接口信息进行管理。

3.1 对象实现技术

根据接口池中接口池对象之间继承和包含的关系,在实现时将接口池设计为树形结构。

实现包含类(Container)时需要解决的基本问题是定义包含类的数据结构,考虑到各 Container 所包含的对象的差异,实现时必须解决如何组织这些对象,使得从 Container 出发可以找到它所包含的全部对象,并且体现出这些接口池对象表示的接口定义间包含关系的嵌套性的问题。

由于除了根对象以外,接口池中所有的对象都是 Contained 类的子类,所以在实现时定义了一个 Contained 类序列 Contents_(所谓序列 Sequence 是 OMG IDL 中的一种类型,实际上是可以动态分配空间的数组),存放相应接口池对象的对象引用,来表示 Container 类对象所直接包含的对象。通过浏览相应的 Contained 类序列,一个包含对象可以找到它所直接包含的接口池对象,这些接口池对象都属于被包含对象(Contained)。对于 Contained 类序列中的每一个被包含对象,如果它还包含有其它接口池对象,那么它也是 Container 类的子类。通过 Narrowing 机制,可以将这个被包含对象转化为包含对象,通过它所包含的对象序列 Contents_,浏览这个对象所直接包含的接口池对象,从而利用包含类中对被包含对象的组织和 Narrowing 机制实现了对接口池对象之间包含关系表示。接口池对象间包含关系用这样一种多叉树形式表示,不仅有利于包含类的对象组织,而且自然地解决了包含关系嵌套性的表示。

3.2 服务程序实现技术

为统一对接口池进行管理,提供对接口池的访问,在服务方实现了接口池服务程序 Irserver,负责生成接口池,并且根据用户的请求,进行相应的事件处理。

服务程序生成接口池时,首先对 OMG IDL 文件进行词法分析和语法分析,创建相应的接口池根对象 Repository;然后用 IDL 分析器遍历 IDL 文件,创建相应对象,依次填充每一个包含对象的被包含对象序列 Contents_,生成对象之间的树形关系。由于在 IDL 文件中定义的服务对象的公共接口是从模块、属性到接口、操作的顺序书写的,所以对 IDL 文件进行词法分析和语法分析,生成相应的接口池对象的过程,实际上就是首先填充底层包含对象的被包含对象序列 Contents_,然后再填充上层包含对象的被包含对象序列 Contents_ 的过程,即先序生成接口池对象多叉树。

当接口池服务程序被启动创建接口池以后,Irserver 持续监听相应的 Socket 端口,接收用户请求,进行相应的事件处理并返回操作结果,直至系统关闭。

4 结束语

CORBA 标准将对象技术和分布处理设计思想结合在一起,完善了分布处理的开放体系结构,使得分布对象技术能够适应新的应用需求,为进一步的集成、共享以及合作问题提供良好的解决基础。

我们接照 CORBA 标准实现的集成中间件 StarBus 是一个屏蔽实现细节、实现透明传输、集成不同用户特长的基于客户/服务器模式、面向对象、开放的分布式计算集成环境,为面向协同工作的集成化分布式客户/服务器应用提供了有效的开发途径,已经在面向大型电子系统并行工程的协同式设计过程管理等系统中得到应用。

参考文献

1 Soley R M. Object Management Architecture Guide, John Wiley & Sons, Inc., 1995

2 OMG. The Common Object Request Broker: Architecture and specification, Revision 2.0, 1995

3 OMG. Interface Repository Object Management Group, version 1.0.2, OMG TC Document, 1995

4 吴泉源, 王怀民, 邹鹏. YHCS 客户/服务器计算机系统研究. 计算机学报, 1997, 20(增刊): 4～9