分布式系统 面向对象 XDCOODS

18-22

部件化面向对象分布式系统 XDCOODS

边平定 蔡希尧

(西安电子科技大学软件工程研究所 西安 710071)

摘 要 In this paper, the object model XDDCOM(XiDian Distributed Component Object Model) is presented. Based on XDDCOM, a prototype system XDCOODS(XiDian Component Object Oriented Distributed System)is constructed which surports object orientation and component ware.

关键词 Distributed Systems, Object Orientation, Component Ware, Object Oriented Distributed Systems.

一、概 述

分布式系统是计算机技术中重要的研究领域。近年来,向下优化(Downsizing)和适度优化(Rightsizing)已经成为一种重要的发展潮流,因此研究由多台自治的计算机经局域网或广域网连接起来的分布式系统,即多计算机系统有重要的意义。

面向对象技术已经被人们广泛接受,在数据库系统、操作系统、分布式系统、系统分析、系统设计等许多方面得到了广泛的应用并取得了许多重要的成果。面向对象技术已经成为计算机技术领域的基础技术之一。

软件开发效率不高是许多年以来一直难以解决的问题,为此产生了各种新技术,软件重用是解决这个问题的重要手段。软件部件(Component Ware)是最新的解决软件重用问题的技术,并且部件化的软件系统易于进化、维护、重构,有较好的动态性能。因此,软件部件化是软件系统发展的主要方向。

将面向对象技术、软件部件、分布式系统结合起来,研究支持软件部件的面向对象分布式系统是解决异构、分布环境下软件重用,资源共享,互操作等问题的重要途径。

本文介绍我们在分布式系统研究中所建造的一个部件化面向对象分布式系统原型 XDCOODS(XiDian Component Object Oriented Distributed System)。XDCOODS 的主要目标是:

编译工作较为模糊;第三 Fortran 语言并行化的工作在高性能并行计算中的作用和工作的延续性,使面向对象 C++语言并行化的研究工作力量不足。

本文重点论述面向对象 C++语言的几种并行模型和并行/分布实现技术,并介绍了我们设计的并行 C++系统 OOCPCS,最后简要地讨论了 C++并行化的几个相关问题。

参考文献

[1] B. Stroustrup, The C++ Programming Language. Addison. Wesley, 1986

[2]高耀清等,面向对象语言 Smalltalk-80 的并行与分布实现技术的研究,计算机研究与发展,7,1993

[3]Stephen S. Yau, Xiaoping, PROOF: A Parallel Object-oriented Functional Computation Model, Journal of Parallel and Distributed Computing, 12, 1991

[4] Arvind and Kim P. Gostelow, The U-Interpreter, IEEE Computer, February 1982

[5]Dennis KAFURA, Manibrata M., ACT++: A Class Library for Concurrent Programming in C++ Using Actor, JOOP. October, 1993

[6] Laxmikant V. Kale et al., CHARM++: A Portable Concurrent Object Oriented System Based on C++, OOPSLA' 93

[7]B. N. Bershad et al., Presto: A System for Object-Oriented Parallel Programming, Software Practice and Experience Aug., 1988

[8]Rajiv Trehan, Concurrent Object Oriented 'C' (cooC), ACM SIGPLAN Notices, vol 28, no. 2, 1993

·支持大型、异构、分布式环境：应能支持大型综合信息系统，企业计算，自动化指挥系统等。

·部件化：在XDCOODS的支持下，应用系统由多个软件部件构成。部件可以随着系统的进化被替换而不影响其它的部件；部件可被其它的应用通过简单的方式重用；部件可以被重新组织以适应系统重构。

·面向对象：充分支持面向对象范型。在程序设计语言一级，尽量保持原程序设计语言的风格、语法、语义不变，如在C++语言中，类、对象、类继承仍保留原来的语义；在运行时系统或操作系统一级，利用对象代理机制的灵活性将异构、分布等屏蔽起来并为语言一级提供支持。

·中立于语言的部件、类、对象：由一种程序设计语言实现的部件、类、对象可被其它语言所用。

·并行处理：不同的部件可以同时在不同的处理机上运行以提高效率。

·资源共享：应用程序或用户能透明地共享系统中的设备、数据、软件等资源。

·安全控制：提供安全控制机制以防止资源被未授权的用户使用。

·容错机制：能利用系统的冗余资源提供某种容错机制。

为了实现上述目标，我们提出一个适用于建造部件化大型、异构、分布式系统的面向对象模型XDDCOM(XiDian Distributed Component Object Model)并在些基础上构造了原型系统XDCOODS，对有关的结构和模型进行验证。

二、对象模型XDDCOM

对象模型定义基本的概念、关系、接口及语义等，是讨论问题的基础。

2.1 部件

在XDDCOM中应用系统由若干部件组合而成，部件是类的容器，是系统的基本建筑单位。部件具有以下特性：

·部件是二进制的，替换或修改系统中的某个部件不导致系统中其它部件的修改和重新编译；

·部件能动态装入，不需要预先启动；

·封装了特定的功能：部件中包含有类，类实现了特定的功能；

·通过消息传递与其它部件交互：与其它部件的交互遵守类接口规定的契约；

·对部件可部分重用：部件的重用是部件中类的重用，XDDCOM支持类的多重继承；

系统中的每一个部件都有唯一的标识，与这个标识关联的有下列属性：1)端口——部件与外界通信的通路；2)权限——权限控制；3)使用计数——目前正在使用此部件客户的数目。

2.2 类

类是面向对象程序设计中最重要的一个基本概念[1]。部件所提供的特定的功能由类体现，每个部件都包含若干个类。一个类由类接口和类实现构成，它们体现了一组对象的共同特性。类是一个模板，能用以生成给定类型的对象。

在XDDCOM中，类由以下几个方面构成：类的名字；和其它类的关系，如继承关系等；类的接口；结构描述；类的实现(行为描述)；

系统中的每一个类有唯一的标识，与这个标识关联的有下列属性：·端口——类与外界通信的通路，类的端口与其所属的部件的端口一致；·权限——权限控制；

2.3 对象

对象是类的实例。对象的结构、行为、对外的接口由它的类定义。在对象的生命周期内，对象只属于某一个类[1]。

在XDDCOM中，对象有唯一的标识，与这个标识关联的有下列属性：·权限——权限控制；·使用计数——目前正在使用此对象客户的数目。

对象具有方法调度机制，当对象收到一个消息时，方法调度机制能决定调用哪个方法。

2.4 类接口(对象类型)、类实现

在XDDCOM中，类分为类接口、类实现两个部分。类接口说明能施加于类的实例(对象)的操作(方法)，包括操作(方法)的名称、参数的类型及返回值的类型。类接口还说明外部可见的属性。类接口也称为对象类型(Object Type)，它定义对象外部可见的属性和能施加于对象的操作。类实现是对类接口所定义的操作的具体的编码实现。同一种类的接口可有不同的类的实现，类接口和一种具体的类实现一起构成一个类，因此，同一对象类型(类接口)可对应不同的类。

2.5 类继承

继承是在类、子类、以及对象之间自动地共享结构和方法的一种机制。子类不仅可以继承父类的行为(操作、方法等)，也可以继承表示(实例变量)。如果父类中的某些行为不适用于子类，则程序设计人员可以重置这些方法，即重写方法新的实现部分。

继承性分为单重继承和多重继承两类。单重继承时一个子类只有一个父类，多重继承时一个子类可以有多于一个的父类[1]。XDDCOM 支持多重继承。

在 XDDCOM 中继承的语义：类继承的语义是沿着继承的路径，把所有的父类的结构和行为（按照作用域规则）进行合成。合成后的类具有其所有父类的结构和行为。

2.6 代理

代理是面向对象程序设计的另一种共享机制[1,3]。采用代理的原型系统中有以下的特点：

- 没有类和实例的区别，任何一个对象都可作为原型；
- 实例可以相互共享，而不是彼此独立；
- 实例的生成只计涉及当前还不能共享其它实例的那些属性结构及行为，和用来描述共享的消息传递，而不是沿着继承的路径展开；
- 实例可以改变与原型的从属关系，在继承机制中这是不允许的；
- 在继承机制中，一个对象可以通过生成它的类或更高的父类继承信息，但控制权是归自己的。在代理机制中，控制权也将传递给代理的对象。

代理机制具有很大的灵活性，在 XDDCOM 中，我们利用代理来实现类的继承机制。

如有两个类 CLASS_A，CLASS_B，CLASS_B 继承 CLASS_A。我们用 C++语言来描述这两个类的结构：

```
class CLASS_A
{
 public;
 void method1(void){method2( )};
 virtual int method2(void){return 1};
};
class CLASS_B:public CLASS_A
{
  public int method2(void){return 2};
};
```

在 XDDCOM 中，利用代理机制来实现以上的继承关系。CLASS_B 的实例 B_OBJ 由分布在不同部件中的两个实例（PART_A_OBJ，PART_B_OBJ）构成，由图 1 所示。

我们称 PART_A_OBJ，PART_B_OBJ 为部分对象。每个对象中包含以下的信息：

- 对象标识：由 this_oid 存储，表示本对象的标识；
- 父对象标识：由 p_oid 存储，表示与本对象关联的父类的部分对象的标识，p_oid 是一个数组，为支持多重继承可存储多个父对象标识；

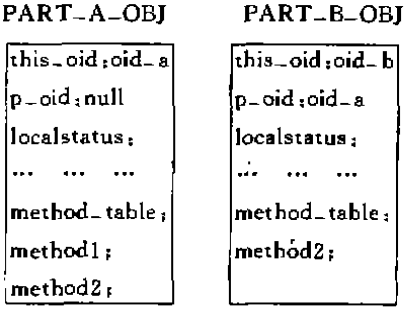

图 1

- 局部状态：存储本部分对象的类接口所定义的外部可见的属性的值；
- 方法跳转表：本部分对象的类接口所定义的操作的实现的入口。

当我们调用 B_OBJ.method1( )时，部分对象 PART_B_OBJ 的方法调度机制首先在本地的方法跳转表中进行匹配，因本地的接口未定义 method1( )，因此这个请求被送到父对象进行处理，同时，PART_B_OBJ 的对象标识 oid_b 作为参数送给父对象，父对象的方法调度机制在本地的方法跳转表中匹配到了 method1( )并执行，method1( )在执行时调用 method2( )，而 method2( )是虚函数并在 CLASS_B 中被重置，因此父对象又请求 PART_B_OBJ 服务（利用对象标识 oid_b），PART_B_OBJ 的方法调度机制调用本地的方法 method2( )，执行的结果依次返回调用方。

由这个例子可以看出，利用代理机制可以很好地解决分布、异构环境下类的继承问题。

2.7 类接口定义语言（IDL）

类接口定义语言是一种高层次的描述语言，用于描述类对外部的接口。类对外部的接口包括操作接口和外部可见的属性。如：

```
//IDL
interface account{
  readonly attribute float balance;
  void makeLodgement(float );
  void makeWithdrawl(float );
};
```

类接口定义语言不是一种完整的编程语言，只用于定义接口而不涉及接口如何实现。类接口定义语言还定义类之间的继承关系。

引入类接口定义语言是为了使类接口的定义中立于任何程序设计语言，然后将类接口定义转换为具体的程序设计语言中对应的接口。这样，一个由

C++实现的类就可被C或其它的语言使用，反之，也是一样。

在XDDCOM中，类接口定义语言采用OMG(Object Management Group)定义的CORBA接口定义语言(CORBA IDL)。

三、XDCOODS的结构和实现

XDCOODS是在XDDCOM基础上所实现的一个部件化面向对象分布式系统的原型。其实现环境是Windows NT操作系统，利用MS Visual C++作为实现语言，通信采用TCP/IP协议。在XDCOODS中，我们全面实现了XDDCOM对象模型并加入了相应的管理功能。

XDCOODS分为两个层次，即运行时系统层和程序设计语言层。在程序设计语言层，对每一种所支持的语言，保留原程序设计语言的概念和语法不变。程序设计语言中的类、对象以及类间的关系都映射到XDCOODS运行时系统中语言中立的类、对象及类之间的关系。如对C++来说，类、对象、类继承等概念和语法与标准的C++完全一致。在运行时系统层，XDCOODS支持语言中立的部件、类、对象，提供具体语言到语言中立的类、对象的联编并对部件、类、对象及类、对象之间的关系进行管理。XDCOODS的结构如图2所示。

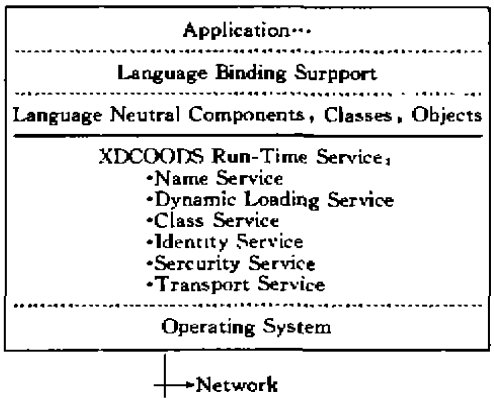

图2

下面我们分别对图2中的内容进行讨论。

3.1 应用程序

对应用程序员来说，在XDCOODS下程序设计分为部件设计和应用程序设计。

3.1.1 部件设计。进行部件设计先要用类接口定义语言进行类接口定义，类接口定义表明部件所实现的类对外部的接口。

对类接口定义文件进行预处理，得到两个头文件，一个为实现部件的程序包含；一个为使用部件的应用程序包含。

部件的实现程序只需为部件中的类提供具体的实现代码。

3.1.2 应用程序设计。如果应用程序要利用某个部件所提供的类，可以有两种方法。

静态接口：是包含由预处理程序为应用程序所产生的头文件，然后就可以象使用自己定义的类一样使用部件中的类；

动态接口：是利用XDCOODS提供的函数动态地装入部件并按应用程序的要求产生某个类的实例，应用程序保留这个对象的标识，并可在需要时请求这个对象为自己服务。

静态接口简便易行，并可利用语言的编译系统进行类型检查，易于保证程序的正确性。

动态接口具有很高的动态性和灵活性，可实现多种多样的共享机制，如代理、聚集(Aggregate)等，其缺点是程序较为复杂，不易控制。

3.2 语言联编支持

语言联编支持为支持具体程序设计语言到XDCOODS运行时系统所管理的类、对象联编的机制。是为支持静态接口和动态接口所提供的一组函数接口，主要的函数有以下几个：

InvokeClass：将包含指定类的部件装入内存并初始化；

RequestObject：请求指定对象服务；

GetOid：请求一个唯一的对象标识；

RegistObject：将一个对象登录到名字服务器中；

DeregistObject：将一个对象从名字服务器中撤消；

对所支持的每种程序设计语言，都有一个相应的库(LIB)提供对这些函数调用的接口。

3.3 语言中立的部件、类、对象

XDCOODS对部件、类、对象进行管理，这些部件、类、对象中立于程序设计语言，即可由不同的语言实现和使用。XDCOODS管理部件的装入、终止，对象的产生、撤消、请求以及访问权限的控制等。

3.4 并发处理与同步

XDCOODS的并发处理依赖于操作系统。在我们的原型系统中，利用Windows NT所提供的多进程和多线程可以实现并发处理及并行处理以提高系统的性能。同步、互斥机制也由操作系统提供。

需要说明的是，在我们的对象模型中，对象是通过端口与外部进行通信的，对象之间的交互是同步

的。这样，外部对对象的所有请求都在端口上顺序化了，因此对象内部不支持并发操作，方便了对象内部的管理。XDCOODS是对象级大粒度并发，不支持方法级的并发。

3.5 容错与异常处理

容错在分布式系统的研究和应用中有着非常重要的地位，其主要思想是利用系统资源的冗余，在系统局部失效的情况下系统仍能正常工作。作为一个原型系统，XDCOODS未提供容错机制。

如果因为系统局部失效或系统资源不能满足要求，XDCOODS会向应用程序返回异常码，应用系统可利用程序设计语言所提供的异常处理机制，按照应用的特点自己实现一定程序的容错。

3.6 XDCOODS运行时服务

运行时服务是为XDCOODS方便、高效地进行管理部件、类、对象以及实现对象请求中介(ORB)而提供的。

3.6.1 名字服务。提供XDCOODS系统中大部分信息的登录、查询、删除等服务，能提供诸如对象所属的类，类所属的部件，部件的状态(是激活的还是非激活的)，对象、类、部件的通信端口，部件所处的位置等信息。

在XDCOODS系统中，名字服务可以是集中式的也可以是分布式的。若名字服务是集中式的，则名字服务不是存在于系统中的每个节点上的，但每个节点都有访问名字服务的接口。

3.6.2 动态装入服务。存在于系统中的每一个节点，如果应用程序所请求的类所在的部件是非激活的(未装入内存)，则XDCOODS请求部件所在的节点的动态装入服务将部件装入内存并进行初始化。

3.6.3 类服务。类服务的主要作用是为系统的应用程序设计者提供整个系统所能提供的类的信息。对程序员来说，可以看见和使用的是类和对象而不是部件，部件只是类和对象的载体。类服务和名字服务一样，在系统中是全局的。

3.6.4 标识服务。在XDCOODS中，部件、类、对象都有全局唯一的标识。标识服务是产生系统全局唯一标识的机制。

标识服务在系统中只有一个。为产生全局唯一的标识，在标识服务中使用了操作系统所提供的服务。产生的标识的值与本节点以太网卡的物理地址(全世界唯一)以及产生时的时间等有关。可以保证只要按此算法产生的标识在全世界唯一。

3.6.5 安全服务。类、对象在XDCOODS中有权限控制。安全服务是在应用程序要使用类和对象时对访问权限进行检查的机制。XDCOODS的权限控制是以存取控制列表(ACL)方法实现的。

3.6.6 传输服务。当应用程序所请求的类、对象不在本地时，需要将此请求及参数传送到相应的节点并将结果送回。传送服务提供目的类、对象的定位，利用安全服务进行权限检查，在不同节点间的通信并对参数和返回结果进行格式变换。XDCOODS利用TCP/IP协议进行通信，网间数据表示采用网络数据表示(NDR，Network Data Representation)。

四、结束语

将面向对象技术和软件部件引入异构、分布式系统是一种新的尝试。通过对对象模型、原型系统实现的研究表明对象模型XDDCOM是合适的，XDCOODS的结构是合理的，特别是在运行时层次上利用代理机制实现类的继承机制是非常成功的。通过应用程序设计实验还表明了XDCOODS能有效地提高软件重用程度，资源共享程度，系统互操作性，可以基本上实现我们前面提出的目标。

XDCOODS是一个原型系统，因此象容错机制、类接口定义语言的预处理等未作考虑和实现，有些部分的实现也作了一定的简化，但这些并不影响对XDCOODS基本概念和结构的验证。

参考文献

[1]蔡希尧、陈　平，面向对象技术，西电出版社，1993

[2]Nicol, J. R. et al., Object Orientation in Heterogeneous Distributed Computing Systems, IEEE Computer, Vol. 26, №. 6, 1993

[3]Lieberman, H., Using Prototipical Objects to Implement Shared Behavior in Object-Oriented Systems, ACM SIGPLAN Notices, Vol. 21, №. 11, 1986

[4]Richard Mark Soley, An Object Model for Integration, Computer Standarda and Interfaces, 15 (1993)