“青鸟”系统中永久对象的实现*

柳军飞 邵维忠 杨芙清

(北京大学计算机科学与技术系,北京 100871)

摘要 永久对象概念的实现为程序员提供了在高级语言级一致地操纵内外存的手段,并使程序之间共享数据更方便和有效.本文讨论了永久对象的描述方式和实现技术,介绍了一个集成化软件工程环境——“青鸟”系统中永久对象的实现方法,并详细地给出了其核心部分——“青鸟”对象管理系统(JB2/OMS)的设计.

关键词 面向对象,永久对象,对象管理系统.

永久对象(persistent object)一词,又译为持久对象,在面向对象技术的相关文献中,指生存期超越相应程序的执行期的那些对象.永久性(persistence)的概念即指对象的这种生存特性,支持永久对象概念的高级程序设计语言被称为永久性程序设计语言(PPL).在高级程序设计语言中直接支持永久对象的概念所带来的益处主要有两点:其一,程序员可以直接利用高级语言来操纵内、外存中的对象,对应用程序员而言,内、外存没有区别;其二,为程序之间共享数据提供了方便.

永久对象的实现主要包括两个方面:永久对象的描述方法和实现技术,前者指如何标识对象为永久的,后者则主要指永久对象的存储及管理方面.

有关永久对象方面的工作在文献[1—18]中已有报告,总的看来这些工作大部分尚处于研究和实验阶段.我们在研制集成化软件工程环境——“青鸟”系统(JB2)的过程中,为了给应用程序员和工具开发者提供尽可能的支持,研究了分布式环境中永久对象的实现问题,并在高级程序设计语言中实现了永久对象的概念[19,20].本文介绍了“青鸟”系统中永久对象的描述方法和实现技术,给出了其核心部分——“青鸟”对象管理系统(JB2/OMS)的设计.

1 永久对象的描述

理想的情形是,在高级语言层不区分临时对象和永久对象,由系统识别和处理.但在目前的技术条件下尚难以实现,因此需要在高级程序设计语言中显式地标识对象为永久的.永久对象的描述方法不仅影响程序员对语言的使用,而且影响系统对永久对象的处理方式和

* 本文1994—10—13收到,1994—12—28定稿

作者柳军飞,1965年生,博士后,主要研究领域为软件工程,CASE环境和软件过程技术.邵维忠,1946年生,教授,主要研究领域为软件工程,CASE环境及面向对象的方法和技术.杨芙清,女,1932年生,教授,中国科学院院士,主要研究领域为操作系统和软件工程.

本文通讯联系人:柳军飞,北京100080,中国科学院软件研究所

效率.

1.1　描述方法

一般而言,可以采用3种方法来描述永久对象:核心转储;显式标识;可达性模型.

核心转储是目前最简单的方法,它允许所有对象均是永久的.当一个程序终止时,正在被使用的所有对象被当作一个单独的存储块保存起来,当需要访问对象时,则恢复整个被存储的块.例如一些Lisp系统[1].

标识永久对象的第二种方法是显式标识对象为永久的,"青鸟"系统即采用该方法描述永久对象.该方法需要程序员显式标识每一个永久对象,在编程中对永久对象的处理直观而灵活,此外,便于永久对象的实现.

可达性模型是指从某一指定的永久对象出发直接或间接可达的对象均是永久的.所指定的永久对象被称为永久"根"(root)对象.如PS-Algol[2,3]的永久性标识.这种标识方法减轻了程序员的负担:建立对象时,程序员不必从一开始就去确定它是否为永久的;对于复合对象的永久标识是隐式的,当一个对象成为永久对象时,它的成员对象也成为永久的.

在实际实现中,往往采用上述方法的组合,如Argus系统[4]综合了显式标识和可达性模型两种方法.

1.2　CASE-C++语言的永久对象描述方法

· 语言简介

"青鸟"系统中的CASE-C++语言[21]主要是供工具开发者使用的一种面向对象程序设计语言,它是C++语言的扩展,同时又保留了C++语言的全部功能.它为工具的组成和输入/输出对象提供了统一的描述、使用和运行管理机制.此外,CASE-C++语言是JB2/OMS的对象定义语言与对象操作语言.

基于C++语言,CASE-C++主要有以下几点扩充:

· 类的共享与重用

CASE-C++语言支持不同用户或工具之间共享类定义信息,并支持类定义的重用.

· 永久对象

CASE-C++语言支持永久对象的定义与使用.

· 对象粒度

CASE-C++语言的对象粒度范围较宽,使用CASE-C++语言提供的对象类,可统一地描述大到由多个文件组成的对象,小到仅由一个属性组成的对象.因此,比较适应于CASE环境.

· 链(Link)

CASE-C++语言提供链机制,链可用于建立对象间的参照关系,也可用于描述对象的结构关系.与C++的指针相比,链具有以下两个特点:(1)它可以是永久的,(2)它可以具有属性,从而使描述能力大为加强.

· 内容(content)

CASE-C++语言允许对象具有存储长度可以动态变化的属性,这种属性称作"内容",特别适于描述被编辑的正文、图形等.

· 永久对象的描述

CASE－C＋＋提供永久对象的定义、存储、检索机制. 永久对象一经建立，若不显式地删除则永久地存储在对象库中，对象库由对象管理系统进行管理. 在赋给相应权限的情况下，其它程序可访问永久对象并进行操作.

定义永久对象时，采用显式声明，仅需在声明时加上关键字 persistent，如：

```
class x{
...
}
persistent x a;
x b;
```

其中对象 a 是一个永久对象，而 b 为临时对象. 当程序结束时，b 不再存在，而 a 则被写回到对象库中.

对于已经存储在对象库中的对象，可利用 retrieve 函数检索它，例如：

```
b=retrieve (perobjname);
```

其中 perobjname 是对象名.

CASE－C＋＋支持正交永久性的实现，即任何 CASE－C＋＋类的对象均可以是永久的. JB2/OMS 为永久性的实现提供运行时的支持.

2 永久对象的实现

实现永久对象的主要任务是如何存储和管理永久对象. 永久对象的存储模型不同于传统的数据的存储模型，前者必须维持其相应的语义；标准性和类属性是永久对象存储所追求的两个目标：人们希望永久对象的存储系统和传统的文件系统一样，有一个标准的存储模型，以减少重复性的工作，增加可移植性以及永久对象之间相互通信的能力，最终取代文件系统；如果能建立一个类属的永久对象存储器，则可为不同的永久性 OOPL 提供永久性实现支持.

然而，在现有的硬件和软件基础上实现标准的或类属的永久对象存储系统是困难的，标准系统的取得需要硬件的支持. 目前，大多数工作零散地体现在系统的各个层次中.

在“青鸟”系统中，对象被区分为临时对象和永久对象，临时对象由语言编译系统本身实现，永久对象通过 JB2/OMS 的支持实现. JB2/OMS 为 CASE－C＋＋提供永久性实现支持.

3 JB2/OMS

JB2/OMS 是“青鸟”系统(JB2)的核心部分，为 JB2 环境中各种工具提供对象存储和管理服务，同时为 CASE－C＋＋语言提供永久性实现支持.

3.1 设计目标

设计 JB2/OMS 的主要目标有如下几点：

(1)永久性：在多次引用对象过程中甚至在系统停机或系统崩溃的情况下必须维持对象的状态. 因此，要求实现永久性存储和管理，此外，必须提供永久性语言的运行支持.

(2)分布性:对象可以分布于网络的不同结点上,对于用户而言是分布透明的.

(3)可移动性:对象可以在不同结点之间迁移,由 OMS 隐式地实现.

(4)并发性:允许不同结点上的多个进程同时访问一个对象,由 OMS 实现透明的并发控制,并维持对象的一致性.

(5)可扩充性:OMS 的接口应尽可能通用灵活,以支持多个永久性面向对象语言的实现.

此外,OMS 应具有较高的性能,公平地处理各种应用请求,减少用户的等待时间,同时具有高可靠性.

上述目标是我们在 JB2/OMS 的设计和实现中决定技术路线和实现细节的基础. 自 1993 年 10 月以来,JB2/OMS 已在一组 Sun3, Sun4 系统工作站所组成的局域网上运行,基本上达到了我们预期的目标.

3.2　永久对象的存储

在 JB2 系统中,我们采取了提供统一的对象模型、统一的 OMS 接口的方式,使 OMS 成为 JB2 系统的一个"中心". JB2 系统在 OMS 的支持下提供一个永久性面向对象程序设计语言给工具开发者和其他应用人员使用,从而有力地支持工具的集成和对象的共享[20].

· 对象模型

JB2/OMS 是操作系统内核的扩充,直接支持面向对象的对象模型. JB2/OMS 所支持的对象模型与 C++ 的对象模型相似,但有如下几点不同:(1)对象可以是永久的. (2)对象之间的关系可以用链(Link)来表示. (3)对象可以带有内容,它提供了表示文件类型的对象的一种方法. 由于 JB2 系统基于 UNIX 操作系统,把文件也看作一个对象,使整个系统在概念上更加统一,而且也方便了用户. 带内容的对象由三部分组成:一部分是对象的属性,一部分是链描述,第三部分是具体的内容.

· 对象存储模型

一个对象由多个物理段组成(图 1):对象描述块(objdesc)记录对象的有关管理信息,类标识(classdesc)唯一地表示该对象的类,objectname 为对象名,pid 为该对象的永久对象标识,是全局唯一的,attrptr 为指向属性块的指针,linkptr 指向该对象的首链的链描述块(如果存在链),contptr 指向该对象的内容,如果没有内容,该指针为空. 链描述块(linkdesc)用以记录该对象的某一条链的链名(Linkname),和所指向的目的对象名(desobjname)等属性.

为了减少存储文件的数量,提高外存访问的效率,对于每一类的所有对象,基于上述分段来组织文件. 在 JB2 环境中的工具或用户程序在使用某一永久对象之前,该对象的相应类必须被编译并安装在全局类树上. 将永久对象调入内存时,内、外存的格式转换是由系统完成的.

3.3　永久对象的管理

· 逻辑结构

JB2/OMS 逻辑结构如图 2 所示.

对象管理层基于 UNIX 操作系统内核,它完成对象的存、取、迁移,以及并发控制、缓冲区管理等功能;运行支持为永久性语言提供一个运行支持函数库,主要包括读、写、删除对象

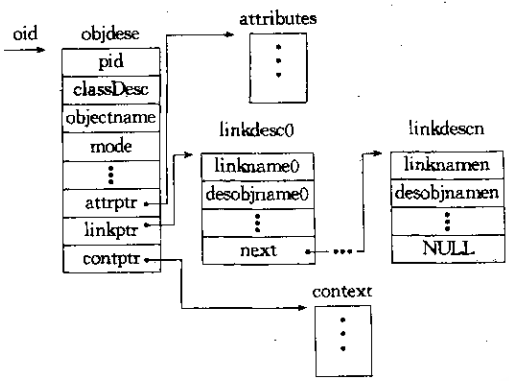

图1 JB2/OMS对象存储模型

图2 JB2/OMS层次结构

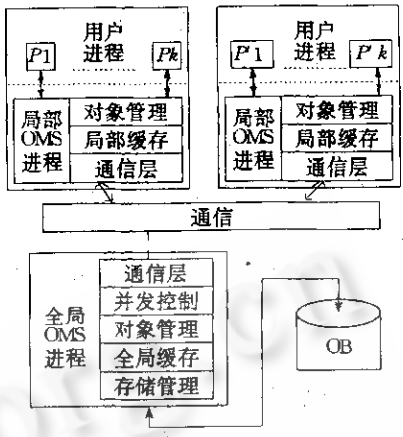

图3 JB2/OMS结构示意图

及链的若干函数，为用户提供永久性、分布透明性的支持.

· 进程布局

JB2/OMS 的进程布局如图 3 所示.

在网络系统中，指定某一结点作为全局服务器运行一个全局 OMS 进程，由它协调全局的并发控制，完成对象的存、取和外存管理，同时，它维持一个全局对象缓冲区，存放系统中最近所使用的对象. 在每一结点上运行一个局部 OMS 进程，它负责接收本结点上的所有用户进程的请求. 完成相应的对象管理功能，同时，也维持一个局部对象缓冲区，存放该结点上所有用户进程最近访问的对象. 当某一用户进程欲访问一个对象时，首先由局部 OMS 进程在局部缓冲区中查找该对象，如果没有找到该对象，则发送消息到全局 OMS 进程，由全局 OMS 在全局缓冲区中查找该对象是否已经存在，如果该对象存在但未被激活，则由全局 OMS 进程从磁盘对象库中读入该对象并送到相应的局部 OMS 进程的缓冲区中.

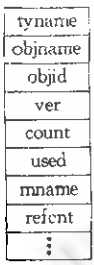

图4 内存对象表的表项

· 对象管理

全局 OMS 进程和每一结点上的 OMS 进程协作，管理 JB2 环境中所有永久对象. 由局部 OMS 进程负责，在每一结点维持一个局部对象缓冲区. 局部缓冲区内的所有对象记录在局部对象表 L—OT 中. L—OT 表的每一表项主要包含如下信息：count：表示当前只读进程的数量；ver：对象的版本号；tyname：对象的类名；used：标志对象是否被使用；objname：对象名；objid：对象标识；同时，全局 OMS 管理一个全局缓冲区，并维持一个全局对象表 OT，OT 表记录目前驻留在所有结点中的对象. OT 表的内容和 L—OT 表大致相同(见图 4)，所增加的表项主要有：mname：结点名，记录相应对象的最新版本驻留在网络中的哪一结点上；refcnt：引用计数.

JB2/OMS 提供的每一个接口操作均是原子的，以保证接口函数操作期间对象的完整性. 系统允许多个用户进程同时读一个对象，但任何时侯只允许一个进程写对象. 并发控制采用主结点封锁法，由并发控制子系统按照用户进程的不同要求给对象加锁，加锁用于整个对象. 同时，OMS 进程管理一个等待锁定的先来先受理的进程队列，选择这种方式的主要目

的是出于减少网络通信的开销,简化并发控制系统的设计.

当用户进程请求访问一个永久对象时,由局部OMS进程和全局OMS进程进行相应的权限检查和定位.为了减少网络通信,大部分工作由局部OMS进程完成.但对于大部分访问请求局部OMS进程必须和全局OMS进程通信,以确定被访问对象的最新版本的位置,并将被访问的对象迁移或复制到局部缓冲区中.

为了防止某些结点出现故障而造成死锁,全局OMS进程实现了一种询问机制,周期性地向所有结点上的局部OMS发送询问消息,如果在规定的时间间隔内未收到某一局部OMS进程的回答,则认为该结点出现了故障.此时全局OMS进程释放被该结点上的所有用户进程所加锁的对象,并记录有关信息.同时设置了异常处理程序,处理用户进程和局部OMS进程的异常情况.永久对象的激活和回收与缓冲区的管理密切相关.此处对象的激活指将永久对象调入内存;而回收有两方面的含义,一种回收是指对象将从内存中被清除;另一种回收是指将该对象从对象库中删除.永久对象只能由具有相应权限的用户显式地删除.我们在实现对象的激活和回收时基于如下两条原则:(1)对象应尽可能保持激活态,即直到主存空间不够且其它对象等待激活时才将某些激活态对象回收;(2)当对象有访问请求时不能被回收,即只有当OT表中该对象对应的refcnt为0的情况下才能被回收.这可能带来一些其它问题.其中一个问题是,所有的激活对象均有访问请求,但此时已经没有足够的对象缓冲区空间来激活其它对象.我们采取的解决办法是,如果一个对象被加读写锁,则将该对象映射到用户进程空间之后,对象缓冲区只保留对象描述块.对于对象缓冲区内refcnt为0的对象,只在空间不够时进行回收.回收算法基于最近最少使用.这是基于如下认识:过去一段时间内最经常被访问的对象,将来对其进行访问的可能性最大.

当某一对象被要求显式删除时,OMS检查相应用户进程的权限是否满足,以及refcnt是否为0,在上述条件满足之后,将该对象从缓冲区中清除.但并不立即在外存上进行存储回收,而是由外存管理过程进行标记,在必要的时候进行回收.

3.4　支持CASE－C＋＋的永久性实现

当用户程序声明和建立一个永久对象时,首先要判断该对象是否存在,如果存在,则将该对象映射到其进程的上下文中;否则,要通知OMS,使被建立的对象为全局可知.因此,CASE－C＋＋编译器在处理每个永久对象声明时插入运行支持函数crobj(),其格式如下:

```
object － id = crobj (classdesc, objname, mode, flag);
```

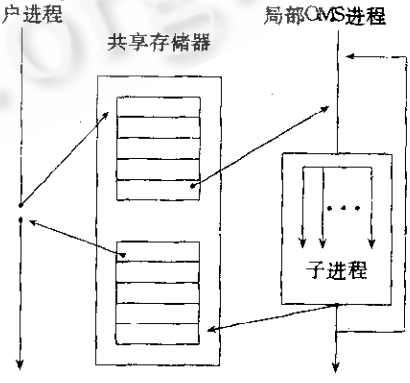

图5　用户进程和OMS进程的通信

通过运行支持接口函数,用户进程向OMS进程发送消息,请求服务.所有用户进程和OMS进程之间的通信利用共享存储器方式实现,如图5所示.这种通信方式效率比较高,但必须严格地保证互斥和同步操作,在设计JB2/OMS运行支持过程中,设计了两个信号量操作原语P(),V()以保证互斥和同步.

目前,JB2/OMS的运行支持还提供如下函数:readobj():读一个对象;wrobj():写一个

对象;copyobj():拷贝一个对象;chobmade():修改对象的权限;addobcont():增加某一对象的内容;crlink():为某对象建立一条链;delink():删除一条链;readlink():读入某对象的一条链;lslinks():读入某对象的所有链;lockobj():给某对象加锁;freeobj():释放加在某对象上的锁;delobj():删除某一对象.

此外,为授权用户进程提供了停止 OMS 进程的接口.同时,为了处理用户进程的异常情况,提供异常处理接口 exit—all(),它捕俘用户进程的异常情况并通知局部 OMS 进程进行相应处理.

4 结 论

在已有的高级语言基础上扩充永久性支持是一条比较有效的途径,基于 C++语言,"青鸟"系统在高级语言中扩充了永久对象的概念,为用户提供了共享对象和一致地操纵内外存的手段,使程序设计简单而有效.JB2/OMS 是"青鸟"系统的核心部分,它支持系统中永久对象的实现,并实现了对象的位置透明、存取透明和迁移透明以及并发控制透明.

参考文献

1 Teitelman W *et al*. Interlisp reference manual. Xerox, Palo Alto Research Centers, 1978.

2 Atkinson M P. An Approach to persistent programming. Computer Journal, 1983,**26**(4):360—365.

3 Atkinson M P *et al*. PS—algol: an algol with a persistent heap. ACM SIGPLAN Not., 1987. 24—31.

4 Liskov B H. Refinement—from specification to implementation, the Argus language and system. In: Lecture Notes of the Advanced Course on Distributed Systems—Methods and Tools for Specification, Institute for Informatics, Technical University of Munich, 1984.

5 Balch P *et al*. Layered implementation of persistent object store. Software Engineering Journal, 1989,3:123—131.

6 Brown A L, Morrison R. A generic persistent object store. Ibid, 1992,3:161—168.

7 Cardelli L, Amber. Technical Report. AT&T Bell Laboratories, Murray Hill, USA, 1985.

8 Cockshot W P *et al*. Persistent object management system. Software Practice and Experience, 1984,(**14**):49—71.

9 Copeland G, Maier D. Making Smalltalk a database system. In: Proc. ACM SIGMOD, 1984. 316—325.

10 Goldberg Adele, Robson David. Smalltalk—80: the language and its implementation, Addison Wesley, 1983.

11 Harrison C, Powell M. A modular persistent store. In: the Fourth International Workshop on Persistent Object System, 1992. 171—184.

12 Tripathi A *et al*. Management of persistent objects in the Nexus distributed system. In: Proc. of the Sec. I-WOOOS, 1992. 100—1005.

13 Meyer B. Eiffel: the language. Prentice Hall, Englewood Cliffs, N.J., 1992.

14 Richarddson J E, Carey M J. Persistence in the E language: issues and implementation. Software—Practice and Experience, 1989,**19**:115—149.

15 Rosenberg J. The MONADS architecture a layered view. In: the 4th International Workshop on Persistent Object, 1992. 215—225.

16 Shekita E, Zwilling M. Cricket: a mapped, persistent object store. In: The Fourth International Workshop on Persistent Object System, 1992. 89—102.

17 Stonebraker M. Managing persistent objects in a multi—level store. In: ACM SIGMOD, 1991. 2—11.

18 Straw A *et al*. Object management in a persistent Smalltalk system. Software—Practice and Experience. 1989, **19**:719—737.

19 杨芙清,邵维忠,柳军飞.永久对象存贮技术研究.电子学报,1994,22(8):1—8.

20　柳军飞. 分布式面向对象技术研究[博士论文]. 北京大学,1994.
21　北京大学CASE课题组. JB2/CASE—C++语言手册. 1993.

THE IMPLEMENTATION OF PERSISTENT OBJECT IN JB2 SYSTEM

Liu Junfei　Shao Weizhong　Yang Fuqing

(*Department of Computer Science and Technology, Beijing University, Beijing* 100871)

Abstract　The implementation of persistent object affords a means of manipulating RAM and Disks in the same way for programmers in programming languags, and makes it easier and more effective for different programs to share data. This paper discusses the description measures and implementation techniques of persistent object, gives the implementation of persistent object in the integrated software environment——JB2 system and the design of its nuclear——JB2/OMS.

Key words　Object—oriented, persistent object, object management system.