73-77,10 面向对象 分布式 OS

(13)

计算机科学1992 Vol.19 No.6

面向对象的分布式OS设计模型

何炎祥 （武汉大学计算机系 武汉430072）

刘栋臣（东北工学院计算机系 沈阳110006）

TP311.12

摘 要

Object-Oriented distributed OS design model is becoming a development trend. This paper discusses the construction and design model of object-oriented distributed OS, including object concepts, object's synchronization, object's capabilities, object-oriented design methodes, and process management, memory management, communication management, device and I/O management, Finally, the differences between object-oriented distrituted OS design model and process-oriented distributed-OS is presented.

一、对象及对象的构成

对象（object）是由封装起来的数据结构及其相关操作构成的集合，是一种抽象数据类型。数据结构描述了数据文件、可执行模块、进程间通信、信箱、目录、应用程序、硬件资源以及它们的控制软件等等，操作则描述了这些数据结构在操作过程中如何变化。由此，计算机系统可看作是对象的集合，而不再是由资源和进程构成。

对象由一个外部成分和一个内部成分构成。外部成分定义对象的用户视图，包括允许用户使用的操作和外部数据类型的定义。内部成分由内部说明、数据、程序、资源管理者等组成，即被操作的实际对象及其实现和表达形式。

一个对象所允许的操作规定了它的特性，定义了它对于外部世界的行为。这些操作及其内部定义的数据结构和计算的结合描

ld scientific, Singapore

[5] Zhou Longxiang, Transaction Management in Distributed Database System C-POREL, Science in China, Series A, Vol.32, No.2, Feb.1989, pp.222-233

[6] 陆汝钤，高全泉，"Tuili文本及使用手册，中国科学院数学研究所，1990，12

[7] 高全泉，Tuili(推理）语言的编译方法与实现技术，软件学报，1991.2

[8] 高全泉，Tuili实现系统里的多推理机结构，计算机学报，1991，9

[9] Gao Quanquan, An Implemented System of Knowledge Inference Language Tuili-Tuili1.1, Research Report of Laboratory of Management Decision and Information Systems, Academia Sinica, MADIS, YB90-0015, 1990.2

[10] 柴兴无，C-POREL的关系基本机器RBM，中国科学院数学所硕士论文，1987，7

[12] 徐建礼，C-POREL中通信子系统CS的设计与实现，中国科学院数学所硕士论文，1988.6

[12] 周为群，分布数据库系统C-POREL的事务管理TM的设计与实现，中国科学院数学所硕士论文1988.6

述了一个对象的例示。对象是作为一个整体而不是其某个部分被引用的（这就是信息隐蔽和封装的概念）。对象中有关于该对象的类型说明，有对象对于系统视图的子类属的规定，还有标识符和关于其过去历史的信息（状态），后者对外部世界是不可用的，主要被对象管理程序用于内务管理[1]。

例如，一个队列（queue）对象只能从一端加入数据项，从另一端取出数据项。它还提供队列"空"和"满"的状态信息及使用队列的操作，如init，append，remove，destroy以及队列类型和大小的信息，这些为该对象定义了用户的外部视图（图1）。该对象对其内部的实际代码和数据结构进行了保护封装，但这些代码和数据本身又可以是一个对象。图1描述的对象可视为一个通用的队列对象，它可为使用者提供建立多个特性、结构、接口、操作及类型都相同但元素不尽相同的队列对象。

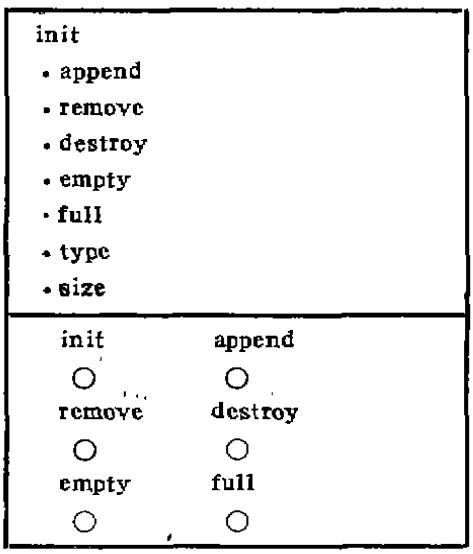

图1 队列对象的描述

二、利用对象模型构造分布式OS的基本方法

要开发实用的分布式OS，需要使用许多简单的对象，即使在一个对象中也可能使用另外的对象。这样，对象本身就成了一种通用部件，可以按一定方式将它们进行各种形式的组合，以提供更强的服务功能。此外，利用对象模型构造分布式OS还要求对众多的对象进行连接，因此，对象模型又是一种构造工具，它提供了方便的机制和策略来实现对象的连接。这表明对象模型有完全不同于进程模型的构造方法。在进程模型中，使用为数不多但很庞杂的进程来构造系统。但在对象模型中，则是将大量的简单对象连接在一起提供服务的。对象模型中的服务比较通用、简便，而进程模型中的服务则比较专用、复杂[2]。

在利用对象模型进行分布式OS设计时，并不需要很特殊的设计技术，采用自上而下或自下而上方法即可。在采用自上而下方法时，设计者先抽象地定义出高级的系统对象，然后根据需要逐步地定义其低级对象。在采用自下而上方法时，先定义出诸如栈、队列、表、树等低级的对象，然后利用这些对象连续地构造出整个系统。据此，设计者只需集中关注对象及其相互作用的操作说明和数据结构的实现，其它则无需考虑。例如，要设计一个调度程序，可用自上而下方法按如下步骤设计：

1）先定义一个处理三级就绪队列对象的高级调度对象（scheduler object）。该调度对象从就绪队列中选择一个对象插入调度队列。

2）然后定义一个低级调度对象（dispatcher object），它按设计的顺序从调度队列中取出对象，将其置成运行态，并请求运行对象为其服务。

3）定义队列对象（queue object）和运行对象（run object）（图2）

这个例子也表明如何通过一些本身具有特定功能但不相交对象的结合来构造OS。

对象的种类很多，常见的是：①由单个进程组成的对象；②由多个进程组成的对象；③由多个进程和共享对象组成的对象；④由多个进程、共享对象及嵌套对象组成的对象。

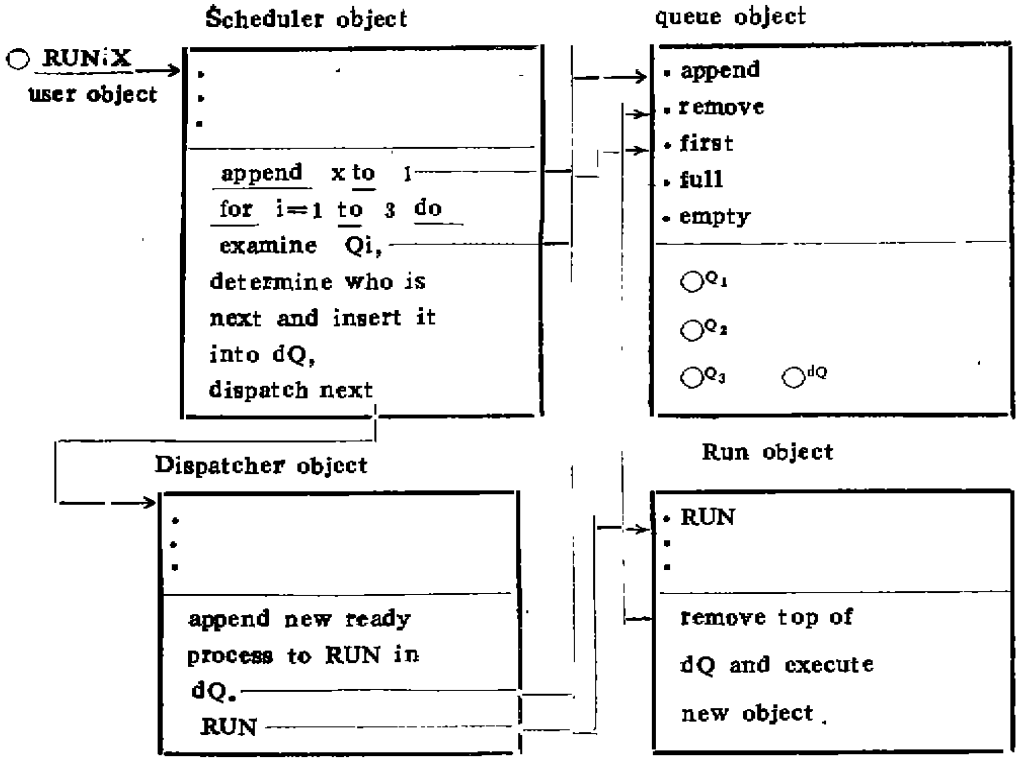

图2 调度对象(合作对象例子)

三、对象的保护域和权限

要保证系统中对象的正确使用，必须对它们进行控制，只允许给予权限的对象去使用相应的服务对象。而且，一个对象在任何时候，都只能存取它当前所要求的资源，这一点可以减轻由于某一对象的故障而引起的系统危害程度。

为了确保这种控制得以实现，在对象模型中引进了"保护域"的概念。每个保护域都定义了一系列对象及每一对象可被该域中其它对象引用的操作。对象只能在一个保护域内工作，而该域规定了此对象的权限和可存取的资源。权限为对象提供了对不希望的用户进行防范的一种手段[4]。

系统中通常有两级对象和权限：

• **用户对象** 可被其它用户对象创建、操作和删除。用户对象可通过从一用户对象到另一用户对象传递权限或通过一公用全局对象连接起来。

• **系统对象** 只能由系统创建、操作、删除和连接。

分布式系统中的每个场点都有一个权限管理员。每个场点的权限管理员只为对象维护本地的权限，但附带一个系统权限目录表，该表用于把对本地的权限请求引导到其所在的场点上。权限封装在请求者不可改变和破坏的对象中，仅权限管理员才有权改变对象的权限。

权限通过名字寻址，该地址由权限管理员解释。当一对象提出某种请求时，权限管理员负责核实该对象是否具有相应的权限，若有，则允许它访问所请求的对象。若请求的对象在另一场点，则本地权限管理员将把请求者交给对应场点上的权限管理员管理。

四、对象的同步

对象的同步可通过对象中类似信号量的操作实现。例如，在对象中引进"signal-wait"操作并加以正确使用，将会为对象模型提供足够高级的同步机制。也可以利用时间戳方法，允许对象将其操作按时间排序，哪个时间早就先执行哪个操作。当然，还可以利用物理锁等方式。

在同步情况下，对象有两个阶段：自治阶段和相关阶段。在自治阶段，对象可相互独立地运行，操作(例如"读")可以并行地完成。在相关阶段，它们的动作可能相互交叉，但操作(例如"写")必须按一定顺序协调地进行。

在分布式系统中，对象的同步有三个主要目标：协调广播、汇聚流串行化和并行流相关。协调广播指的是对多个服务对象的起动序列的同步。这种同步的关键是必须使所有服务对象的操作都初始化并完成。这是一种松散的同步，象在Forth中的操作一样，所有的对象都执行，但并不指定特定的顺序。汇聚流串行化指的是请求对象用与其相关的服务对象对应的已知优先级关系来引用它们的同步策略。这类同步是汇聚的，即多个场点上的计算可能需要相聚在一起，以便另一个计算能继续下去。这些计算假设是互不干扰的，即一对象恰当地执行对某些对象的操作而不影响另一个对象，反之亦如此。并行流相关指的是一些并行执行的对象请求同一对象提供服务时的同步策略。

五、进程管理

进程管理由核对象和进程管理对象联合承担，它们共同完成对各场点的权限管理、对象的创建、操作和删除，以及对象的同步、通信和调度。

核对象是实现分布式OS的基本机构，其主要功能为：

①动态地创建、操作和删除对象。创建是把对象引入系统，并将这一信息通报给有关对象，为其设置存取级别并提供必要的服务，以保证其正常工作。创建活动要求创建者规定被创建的对象驻留在何处、它的状态以及对其操作所要求的其它信息等。

②提供同步原语支持同步。如，提供基于信箱的原语、信号/等待原语，顺序化策略等。

③提供通信原语支持通信。如提供同步或异步通信原语。

④提供调度原语支持调度。核对象必须提供能一致和完全地调度对象的机制。

若核对象已提供以上原语，则进程管理对象具体完成下面的任务：
- 对象的创建；
- 对象操作间的同步；
- 对象的调度；
- 低级调度(dispatching)；
- 对象的删除；
- 对象之间的通信。

进程管理对象接收如下的操作请求命令：

Run x	Send x, y
Create x	Receive x, y
Abort x	Associate x, y
Block x	Fork, Join

处于运行、就绪、等待和阻塞状态的对象必须使用这些命令来执行要求的任务。进程管理对象接收请求对象发出的请求，完成诸如创建一对象、运行一对象、终止一对象、阻塞一对象、对象的分叉与汇合等任务。为完成这些任务，进程管理对象使用核对象提供的原语来控制基层的硬件。例如，要运行对象x，进程管理对象需要做下面的工作：

a) 确定x是否在内存；

b) 若不在内存，则在辅存中找到其位置，为其分配内存空间，将其全部（包括权限等等）装入内存适当位置。

c) 为系统对该对象进行调度提供必要信息。

d) 当调度到它时，核对象将其置为运行态。

上述过程与进程模型中进程管理的处理过程类似，主要区别在于：被作用的对象本身含有其所需的全部状态信息，这些信息不必存放在类似"进程控制块"或其它与该对象分离的数据结构中。

六、存贮管理

存贮管理采用分散式方式，每个场点有一个存贮管理对象，它为来自于进程管理的请求分配和回收存贮空间，以控制整个存贮器需求的变化。整个分布式系统的存贮空间由多个子空间组成，每个存贮管理对象拥有

其所在处理机的那一部分。在为一对象分配空间时，存贮管理对象首先检查其管辖的存贮空间，若可以满足，就为其分配所需空间，并调整现有的存贮区。对外界而言，存贮管理对象只具有分配和回收两个操作。其实，存贮管理对象还可利用其内部的对象，如"无用区收集"(garbage collect)和"紧缩"等对象来协助完成自己的工作，这就自然地形成了对象的嵌套结构。

七、设备管理

物理设备由一层软件（驱动程序）包围着，这层软件为外部世界提供了该设备的视图。设备管理也采用分散式方式，每个场点上都有一个设备管理对象，它主要负责管理本场点上的物理设备，同时附带一个包括系统中所有场点上的物理设备对象的目录表，使得每个对象用其局部观点就可以访问分布于整个系统中的物理设备。当一对象请求访问某一物理设备时，就向本地管理对象提出请求，若本地设备可以满足这一请求，则满足之，否则把该请求发送给系统中另外的设备管理对象来为其服务。这些对用户是透明的[8]。

八、I/O管理

I/O管理由I/O管理对象承担，它主要是对请求对象和设备服务对象之间的交互作用进行控制，为分布式系统中的文件和信息提供信息转发功能。它是文件、设备等对于实际占有它们的使用对象的视图，为这些使用对象提供读写信息的接口。它接收来自服务器对象的中断类操作，提供对输入/输出数据、打开和关闭文件、设备、块等的操作。它还提供一些条件操作，如wait，block，signal等，这些条件操作允许用户对象在读写文件时或者以活动方式等待，或者阻塞、进入睡眠态，或继续其运行，并在操作完成时给出完成信号。这些条件操作为用户对象提供了各种方式的、与I/O操作同步的方法，用户对象可以用它们来实现相互之间的松散同步。

九、通信管理

通信管理负责为分布式系统中合作的对象提供场点间和场点内的通信能力，并为消息选择适当的路径。通信管理提供send,receive，reply，request等通信原语供选用。send原语允许一对象向分布式系统中的任何对象发送消息。receive原语使引用者能够接收来自于分布式系统中任何对象发来的消息。reply原语使引用者对来自另一对象的请求作出响应。request原语为对象提供了请求某一特定服务的表达方式。

根据实际环境的需要，通信管理可以以"发送/接收"为基础提供服务，也可以通过端口、信箱、通道等来实现。前者要求用户知道与之通信的对象名，而后者要求用户只需知道与之对应对象相关的端口名、信箱名或通道名。例如，若通信手段建立在端口～对象类的机构上，则有如下情形：

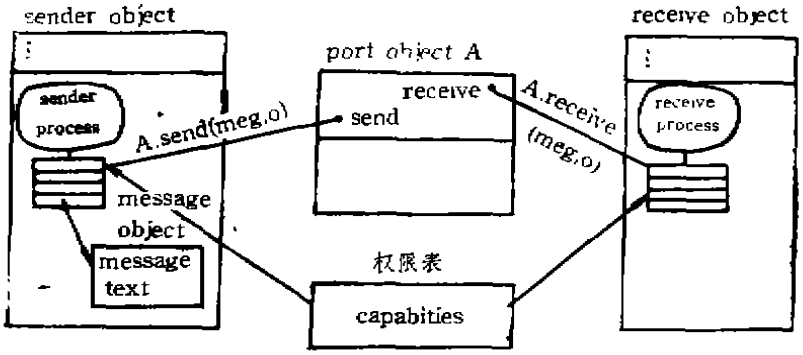

图3 基于端口～对象的通信模型

(转第10页)

神进行研究的。他们从实际例子入手，测试考察每一个逻辑方法，看它能够解决什么问题，而哪些问题又是它不能解决的，使人们清楚地了解到传统的和现有的方法的局限性，为新思想、新方法的产生创造了条件。其次是它使我们的思想不局限于只考虑一个问题，一类问题的解决，而是要考虑到在模拟人类推理的活动中，可能会出现各种各样的问题，除了已经注意到的信息可能是不完全、不确定、不精确、与时间有关和存在例外这些问题外，还会有新的问题出现。它告诉我们，不要满足于原有的思想和方法，需要不断地探索解决问题的新思想、新方法。

事实上，根据这项研究的结果，我们可以构造两个索引文件。一个是关于方法的，它告诉我们，目前在AI领域中，常识推理的模拟都使用了哪些逻辑方法，每一种方法能够解决哪些问题，最适合处理什么问题，满足哪些性质，有什么缺点不足等。另一个是关于问题的索引文件，它包括这样一些信息，在模拟人类推理的过程中，有哪些需要解决的问题，哪些是已被现有的逻辑方法很好地解决了，哪些尚未解决或尚未圆满解决，对于一个给定的问题，有几种不同的解决方法，哪种方法的效果更好一点等。当然随着研究的深入发展，这两个文件的内容也会不断地扩充和修改，但是，在当前它们提供的这些信息无论对逻辑的创造者还是应用者都是非常重要的。简而言之，这项研究的价值在于它的启发性、指导性和示范作用。

最后，我摘录了现代著名的数学家、逻辑学家和科学哲学家A. N. Whitehead的两段话，"……为了学到知识，我们必须首先使自己不为其所拘泥。""……我们应当使我们的体系和系统保持开放。换言之，我们应当对体系和系统的局限性相当敏感。"这就是我推荐这篇报告的目的。

参考文献

[1] Léa Sombé, "Reasoning under Incomplete Information In Artificial Intelligence", International Journal of Intelligent Systems Vol. 5, No. 4, Sep. 1990

（接第77页）

对象按发送者、接收者、端口和消息进行分组，而它们则通过相关的权限连接起来。这种情形的典型模型如图3所示。图中展示了一个发送对象、一个接收对象、一个端口对象、一个消息对象，以及接收者与发送者进程通信的模式。

此外，基于对象模型的分布式OS还包括时间管理、故障检测和故障隔离服务等等。

本文扼要介绍了面向对象的分布式OS的组成及其设计模型，目的旨在概要给出这类系统的构成方法和工作原理，并未涉及实现细节。文中描述的对象模型中，对象是封装起来的一个或一组进程及相关数据结构的集合，是一种抽象的数据类型。可以通过对其操作使用它们，并通过权限及其语义形式控制它们。

任何操作系统都可以看作是由几个基本的管理模块组成，即：进程管理、存贮管理、设备管理、I/O管理及通信管理等，分布式OS也不例外，本文仅从面向对象的观点出发讨论了它们的工作原理及基本性能。

面向对象的分布式OS设计模型与面向进程的分布式OS设计模型的区别在于：

- 对于后者，OS是进程的集合；对于前者，OS是对象的集合；
- 对于后者，用户进程和系统状态的同步与控制通过消息传递实现；对于前者，这种同步和控制是通过权限的管理和分配完成的。

由于对象的概念把数据和处理数据的过程组合成一个整体，它既可以象数据一样被处理，又可以象过程一样去描述处理的过程，因此，可以预料，面向对象的分布式OS设计模型将成为分布式OS的发展方向[5]。（参考文献略）