计算机科学 1995 Vol. 22№. 5

面向对象语言 C++ 并行/分布技术的研究及实现

肖 侬 卢宇彤 胡守仁

（国防科技大学计算机系 长沙 410073）

摘 要 The paper first introduces C++ language, parallelism and some parallel models. Then, the emphasis is put on the design and Implementation of C++ parallel/distributed systems.

关键词 Object, Dataflow, Functional, Paradigmproxy

C++[1]语言是混合型面向对象的程序设计语言，几乎遍布计算机的各个领域。分布式系统和多处理机系统的发展，并行处理技术的应用领域日益扩大，对具有面向对象特性的 C++ 语言并行化就具有重要的意义。本文主要讨论面向对象语言 C++ 的并行/分布技术的研究和实现。

一、面向对象语言的并行性

面向对象语言具有对象、模块化、类、继承、多态性和动态聚束、消息和方法等特性，C++ 是在 C 的基础上融入了面向对象技术的一种语言，它支持以上概念特性。

面向对象程序设计语言的并行性表现在两个方面，其一是对象可以单独执行，并通过同步或异步协议与其余对象和环境通讯，对象之间可以并行执行，称为对象间的并行性；其二是一个对象可以有多个入口，可能具有可并行执行的多个线程。另外，对象内含有子对象，子对象之间可以并行执行，这种并行称为对象内的并行性。

对象之间的通讯方式可分为八种[2]，粗略地可分为同步（请求对象等待结果返回才允许继续执行）、异步（请求对象无需等待即可继续执行）和异步 FUTURE（请求对象无需等待即可继续执行，直到确实需要结果再等待返回值）三种方式。

二、面向对象 C++ 并行模型

C++ 并行模型根据对象活性、继承机制、对象通信模式、对象类型化、对象的层次、对象内的范型几个方面的不同而不同。我们描述几种主要的 C++ 并行模型。

1. 面向对象函数式计算模型

面向对象语言内在的并行性开发受到了传统程序设计语言的处理方式的限制，函数式程序设计方式简单和有效，并且它能开发出细粒度的并行性，为并行计算提供巨大潜能，但由于纯函数式范式缺乏历史性记载，它的表达能力受限。面向对象范式和函数式的结合模型，发挥了各自的优势，一个对象有自己的局部数据和方法，局部数据保证了方法的历史记载，各方法是纯的 applicative functions，基于这种模型的系统既具有面向对象属性，又具有函数式开发并行性的能力，例如 PROOF[3]。

2. ACTOR 模型

Actor 模型将传统的基于对象的概念与函数程序的概念统一起来，是一个基于对象、内在并行性的模型。一个 actor 是一个带有通讯地址和行为的对象，actor 之间可以通过通讯地址异步发送消息，导致在并行上的虚拟开发，每个 actor 带有一个消息队列，必须定义一个 Replacement 以处理向它发送的下一个消息，例如 ACT++[5]。

3. 客户/服务器模型

这类模型将对象看作为客户（Client）或者服务器（Server），引入全局共享对象以便于各个客户程序共享信息。一个对象向另一个对象发请求，通过 message-passing 实现。客户对象向局部或远程服务器对象以同步或异步方式请求服务，服务器对象处理完请求后返回结果。这种模型系统大都用于分布式计算环境。

4. 面向对象的动态计算模型

肖 侬 博士生，目前从事面向对象技术和并行分布式系统的研究。卢宇彤 讲师，目前从事并行分布式系统的研究和实现。胡守仁 教授，博士生导师，从事体系结构、人工神经网络、面向对象技术和并行处理的研究。

动态模型描述了一个并发活跃对象的系统各操作间控制、相互作用和顺序性，强调系统随时间变化的动态控制，而不注重各操作具体做什么、对什么操作和它们如何实现的。动态模型的主要概念是事件和状态图，一个状态图代表带有重要动态行为的类，将它引入到面向对象的计算中，形成融入各种复杂控制关系的对象。这种模型实质上将对象间或对象内并行活动的约束关系，引入到类和对象两个概念层次，以达到并行活动在控制下顺利进行的目的，例如 Concurrent C++和 CHARM++[5]。

5. 面向对象的数据流计算模型

面向对象函数式模型只说明和利用了类和对象的方法函数式的特点，开发了对象间及对象内的并行性，但并未给出对象间各个方法之间的动态联系如何实现。Actor 模型是一个较好的基于对象的内在并发模型，但它缺乏继承性，并且这种模型和传统程序设计语言的设计方法和思想相差甚远。动态模型将并发活动的约束关系引入到类和对象中，虽然并行性在多种层次得到开发，但程序结构不良，增加了系统的复杂性和用户的负担。

我们设计了一个基于面向对象的数据流模型系统。数据流方式表示了整个程序的构造和执行过程，按 U-INTERPRETOR[4]说明，它可以开发程序中的所有的并行性。面向对象的数据流计算模型中的对象方法具有函数式语言的特点，对象之间通过消息传递实现通讯。各对象的请求按照数据流方式连接，一个对象请求的处理结果的输出可能是另一个对象请求的输入。具有这种模型的系统能开发出程序中较大的并行性，例如 OOCPCS。

三、面向对象语言 C++并行/分布式系统的实现

C++是 C 语言的超集，它的并行/分布系统大多使用了原有的 C++或 C 的编译器。

1. 并发 C++系统

并发 C++系统实现于单机或共享主存的多机系统上，对象间采用同步或异步消息通讯，多个对象并发执行。这类系统的并发性主要表现在一个共享的地址空间上的并发，多机共同拥有一个调度器。

(1)基于线程并行。一些系统为并行程序预定义了一些结构，这些结构以类或类型的形式实现，用户可以定义新的类或对象以继承和拥有预定义结构的属性，实现并行活动间同步、互斥、共享及并发，但继承性却受到限制。

Presto[2]提供一套预定义对象类以方便用户编写并行程序。所有对象执行在各处理机共享的一个地址空间上，基于线程并发。对象的属性封装在对象内，对外界透明，对象本身不说明它是同步还是异步激发，对象的使用者也不知道对对象的请求是并行还是串行完成。Presto 的线程库提供了 spinlock、stack creation、starting threads、wake up、living 等机制结构的定义，所有的处理机共同拥有一个对象调度器。

uC++将执行元素的特性组合起来而形成的四种抽象 Corountine、Monitor、Corountine-monitor 和 Task，以新的类型说明符引入到 C++语言中。用户根据这些类型开发程序的并行性，uC++程序由预处理器翻译成嵌有运行库调用的 C++或 C 语言。

(2)对象和进程的合一。CHARM++将对象和进程合一，引入并发对象、重复对象。对这两种对象的调用是异步消息驱动的，对象每一时刻只能处理一个消息。重复对象创建时，在每一个处理机产生一个分支对象进程，共同拥有单一的全局名，它实现了数据并行。

同时，系统还有通讯对象、共享对象、顺序对象。通讯对象是并发对象间的消息实体。共享对象是供各个进程对象共享的信息，有只读对象、仅一次写对象、累加对象、单调对象、分布表，它们拥有系统预定义的操作供用户使用，每一个对象并不一定驻留在一个处理机上，可能是一个分布数据结构。

CHARM++在运行初始时给各进程对象和各对象入口名赋予全局唯一的索引标志号，以支持并发对象、重复对象的多重继承、动态聚束和重载。系统从系统消息缓冲区中按策略选择消息，并根据类型处理它们。

(3)对象和 Actor 共存。C++中的对象是被动的，不是自主活跃的，只有等待外界请求激发才能执行，而 Actor 是自主活跃的，可以主动地选择满足某种条件的某类消息。

ACT++2.0 提供了预定义的 Actor 类，所有的对象都是这个类的实例。类 Actor 表示处理异步消息的 Actor 对象，一个 Actor 有两个成份：一是消息队列，存放输入消息和通信方式；二是它的行为，它是类 behavoir 的实例，一个请求 Actor 的消息的执行是类 behavoir 一个特殊类型实例对象的方法的执行，所有 behavoir 的对象是用户定义的系统预定义类 behavoir 子类的实例。

ACT++系统有两类对象，一类是通常的 C++对

象，另一类是 Actor 对象，发给 Actor 的消息具有异步语义，发给通常的 C++对象具有同步语义。

在 Actor 的计算模型中，用行为的概念代替状态的改变，当 Actor 执行当前的行为时将指明它的替换行为，即它将如何处理下一个消息。ACT++2.0 还提供了 Behavoir sets 用来选择消息，使用 CBOX 机制缓存回答消息(CBOX 是预定义类 CBOX 的实例)。此外，ACT++允许定义异步 I/O 对象支持并发 I/O。

2. 分布式 C++系统

分布式 C++系统允许用户共享网络的资源，这些资源以对象表示。

(1)分布环境下的分布共享对象管理的方法。分布共享对象管理器管理分布式环境下共享的对象资源，对共享对象资源采用一个全局统一的逻辑空间命名或在一不连续、不完整的逻辑空间上命名方式，用以区分任一共享的对象。

这类系统采用代理对象机制，支持分布对象的物理地址的透明。无论用户引用本地还是远程的对象，对象名总是指向一个本地对象的指针，在一结点上任何一个远程对象的引用，都是通过在本地引入一个代理对象来实现访问。代理对象对用户是透明的，它是用户和远程对象之间的转送中介，是对象管理器的私有数据(见图 1)。

一个远程对象的访问也可以通过对象拷贝或对象迁移实现。由于对象频繁产生、消亡、访问，对象拷贝增加系统管理的复杂性及数据一致性的维护，因此，一般采用对象迁移的方法。这种方式又有两种策略，一种是对象在任何状态下允许迁移，开销过大；另一种是对象处于一个请求处理前和处理后的状态可以迁移，正在处理一个请求时不能迁移，开销较小；后一种方式较常用。

当一个对象迁移时，它在所有机器上的代理对象由对象管理器修改。

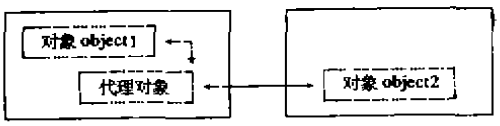

图 1　代理对象的中介作用

(2)进程对象的方法。cooC[8]系统是一个基于进程和对象模型的分布合作计算系统。对象采用了 CLIENT/SERVER 观点发送、接收、处理和回答消息，对象间以异步 FUTURE 方式通讯。一个对象由内存、隐藏函数、一个控制接口和进程组成，和进程成为一个统一体。对象代表进程的边界，并承担起共享资源的保护。每一进程对象通过继承 OBJECT 类拥有一局部控制系统，控制对象内的活动。系统使用 EXCLUSIVE METHODS 方式定义对象控制接口，允许多个请求的并发执行，并保护了共享资源。对象内的并发线程通过策略在一个进程内并发执行。

cooC 系统由对象空间、RUNTIME 空间、网络和存储空间三个软件结构层次组成，支持多用户的合作分布计算。

3. 并行 C++系统

我们设计了一个基于面向对象的大粒度数据流模型的并行 C++系统 OOCPCS，适用于分布式系统下完成一个用户的任务。

OOCPCS 的并行模型是在传统的大粒度数据流模型上引入面向对象技术建立起来的，它支持面向对象的程序设计范式和数据流并行机制。OOCPCS 的数据流 Graph 中的每一个结点都是大(中)粒度执行计算体，引入了对象概念来支持信息共享和并行处理，在并行对象的请求之间动态地建立数据流的关联。OOCPCS 有两类并行对象，一类对象是处理机对象，另一类是状态对象，对这两类对象的请求是异步 FUTURE 方式调用。

处理机对象是一个无永久状态的对象，每次对这类对象的请求形成一个并行处理机实体，它和传统的数据流图中的结点完全等价，只要它所要求的 Tokens 全部匹配，就可以异步地点火执行。它的输出产生了新 Token，流入到数据流图中。

状态对象是一个保持内部状态的对象，供各程序间共享信息。每次对状态对象的请求形成一个该状态对象的并行实体。它和传统的数据流中结点不同，它的点火执行不仅依赖于所连弧 Token 的匹配，还取决于它所对应对象的状态。当请求状态对象时，并行实体就被异步产生，但未必马上执行。一个状态对象可能存在多个并行实体，但状态对象每一时刻只允许一个并行实体进入执行，并且并行实体以产生的次序执行，以保持状态对象的完整性、唯一性和有序性(见例 1 和图 2)。

例 1　一个并行 C++程序

```
CLASS A obja;   /并行状态对象*/
CLASS B objb;   /*处理机对象 */
i=obja. opl( .);    /*产生一个 state actor*/
j=objb  opl(I);     /*产生一个 processor actor*/
obja  op2( i, j);   /* 产生一个 state actor*/
```

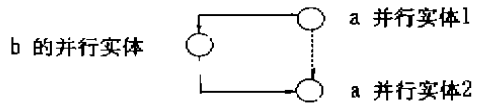

图 2　例 1 的数据流并行图

（虚线表示 a 并行实体 2 在 a 并行实体 1 之后执行）

OOCPCS 系统有全局的状态对象名空间，给一个状态对象一个 ID 号，处理机对象有一个局部 ID 号。系统允许并行对象的迁移，但迁移必须在对象处理一个请求之前或之后进行。

OOCPCS 在用户程序中采用代理对象技术，以控制处理机并行实体的产生和状态对象的并行实体的产生及有序执行，并支持并行对象的物理地址透明性。OOCPCS 使用并行编译技术识别两类并行对象，并自动构造并行模型和优化程序。同时，OOCPCS 的预编译器构造一个类表和类选择器，以及为每一个类建立方法表，每一个方法对应一个索引号，以供系统确定对象请求的对应方法，支持了多重继承、多态性和对象指针。

OOCPCS 是一个为用户提供了对串行程序或加注并行程序并行处理的分布式环境。

四、C++并行化的几个相关问题

面向对象 C++语言并行化涉及到并行处理的许多关键问题，本文简要地讨论面向对象 C++语言并行化中有关面向对象特性的几个问题。

· 继承性。面向对象的继承性使代码共享和重用成为可能，但是继承和并发是不一致的，这表现在并发要求的独立性和继承要求的结构共享之间的矛盾上，其根本原因在于模块化的目标和共享是不兼容的。模块化要求系统各部分严格分开，而共享则希望融和。多重继承则会引起更多的冲突，它的处理更加困难。面向对象语言会产生继承异常，即父类和子类的同步限制发生冲突。一些 C++并行语言虽支持继承，但它们的同步限制是不可继承的，它们通过临界区、消息队列、行为抽象及 enabled set 等结构来表示同步限制。有一些语言采用划分状态和行为集的方法以解决此问题，但效果受限，并且使语言的设计使用复杂化。另一些系统把此问题留给用户负责。

· 并行类库的构造及利用。C++的并行语言由于继承异常使并行类库的建立和使用有一定困难，并且每个并行语言拥有一个自己的并行类库，兼容性差，使面向对象的代码重用的特性受到限制。并行编译器要对程序进行并行识别及数据相关性分析，以抽取并行性，因此有必要对串行类库中的函数、结构等重新加以分析，这就导致了库的重新编译。C++库应当找到一种良好的组织方式，提供一些信息供并行编译器使用而无需对库重新编译，其它语言也存在这样的问题。

· 指针和结构。C++语言拥有大量的指针和复杂结构，对指针和复杂结构进行并行分析、处理是很困难的，这主要由于它的不确定性。这种不确定性在静态上无法识别，在动态情况下无力且无法承担起确定它们具体内容的开销。大多数 C++并行语言避开此类问题，而对于 C++并行编译器，由于指针和复杂结构的不确定性导致数据相关性分析及并行对象分析的不确定性，损失了大量的并行性。

· 类的共享和容错。在分布并行环境下，类继承使用户级上共享代码成为现实，但在系统级上却难以实现，究其原因是在现行的机器和操作系统中对象是以单一、连续、封闭空间地址的进程或进程中的线程来实现的。尽管部分系统在系统级上实现了类继承，但实现方法往往不自然，开销较大。在分布并行环境下所涉及的主要问题还有：各个类分布到系统中的各个结点，如何追踪它的所有双亲，如何实现延迟约束；容错问题如何解决，即当一个系统丢失某些资源时，子类所继承的父类可能消失；一个类的对象如何在系统实现中仅共享各操作，而不共享其状态（每一个对象状态不同）。

五、结束语

面向对象技术广泛应用于语言、操作系统、分布式并行处理系统、数据库等各领域。目前，已出现了许多面向对象并行（并发）C++语言系统，但大多数这类系统主要集中在提供一个良好的并行 C++语言文本和并行分布式支撑环境的研究上，对面向对象语言 C++的并行编译技术的研究较少。这可能有以下几个原因：一是面向对象技术缺乏一套完整严格的理论形式支持；第二是由于面向对象技术应用时间不长，对面向对象语言在什么样的层次上做并行

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