一种基于 MPI 的并发面向对象模型*)

鲁宏伟　肖永玲
(华中科技大学国家外存储专业实验室　武汉 430074)

A Concurrent Object Oriented Model Based on MPI

LU Hong-Wei　XIAO Yong-Ling
(Huazhong University of Science and Technology, Wuhan 430062)
(National Specialized Laboratory - Storage Systems, Wuhan 430074)

Abstract　Object-oriented model possesses inherent concurrency. The integration of concurrency and object-orientation is a new prosperous field. MPI is a message-passing standard and has been adopted by more and more people. This paper proposes a concurrent object-oriented model based on MPI in which asynchronous message-passing is adopted between concurrent objects. This paper simplely introduces the features of the model and design and realization.

Keywords　Message-passing interface, Object-oriented, Asynchronous message-passing, Collective communication, Concurrent class library

1. 引言

面向对象程序设计方法是当今最有前途的软件设计技术之一。面向对象方法是与现实模型相对应的，而现实模型中的对象是并发活动的，因此面向对象方法被认为具有潜在的并发性。将面向对象技术和并发技术结合起来的并发面向对象技术是近几年才兴起的，是一个比较新的研究领域。近年来，国内外提出了许多并发面向对象模型，文[1]提出了 Actor 模型，在该模型中，对象被称为 actor，它是自含的、交互的和独立的软件构件，各 actor 之间通过异步消息传递进行通信。一个 actor 由两部分构成：信箱和行为，信箱用于存放接收到的消息；行为用于对消息进行处理。在处理 1 个消息时，1 个 actor 可以做 3 种操作：生成新的 actor，发消息给熟人和指定一个替代行为。Actor 模型的一些思想对后来的一些并发面向对象模型产生了很大影响，很多较有影响的并发面向对象模型(如：ABCL/1、ACT^{++}等)都是在 actor 模型的基础上进行改进得到的。Actor 模型一般实现在共享内存和紧耦合的多机系统上。actor 模型实现起来比较困难，而且不支持继承。文[2]所提的模型包含主动对象 CONCURRENCY，对象之间采用远程过程调用 RPC 机制进行通信。这种方法也是可行的，但开发高效可靠的并行支撑系统的工作非常浩繁。文[3]提出的模型使用 PVM 做底层支撑系统，包含主动对象 pv_units 以及在它们之间的异步消息传递。该模型最大的特点是提出了虚电路 v_circuits 的概念，通过虚电路在不同的并发对象 pv_units 进行很好的协作。它实现起来也很复杂，而且有一些限制，如它不能处理并发对象的出错，不能识别死锁等。

MPI[4](Massage Passing Interface)是为了统一不同的 MPP 厂家的消息传递的 API 而制定的工业标准。为了制定消息传递标准，1992 年 11 月，正式成立了 MPI 论坛，该论坛吸收了大多数并发计算机厂商、许多欧美大学和研究机构以及并发程序开发人员，大约 40 个组织的 80 多人。从 1993 年 3 月，开始制定 MPI 标准，即对消息传递接口进行标准化，于 1994 年颁布了 MPI-1。MPI 吸收了当今几乎所有的消息传递系统的特色和优点，日益受到并发库开发人员和广大并发程序设计人员的青睐，成为消息传递的标准。MPI 最大的优点是性能，MPI 提供了丰富的点点通信函数，集合通信函数，允许用户自定义消息数据类型，虚拟拓扑结构。MPI 提出的通信控制器的概念提供了对有效、安全、可移植的健壮的并发库的支持。

基于 MPI 的诸多优点，我们提出了一种基于 MPI 的并发面向对象模型，该模型充分利用 MPI 所提供的丰富、高效的通信函数，采用类库方法实现，既具有较高的性能，又具有可扩展性、可复用性。

2. 基于 MPI 的并发面向对象模型

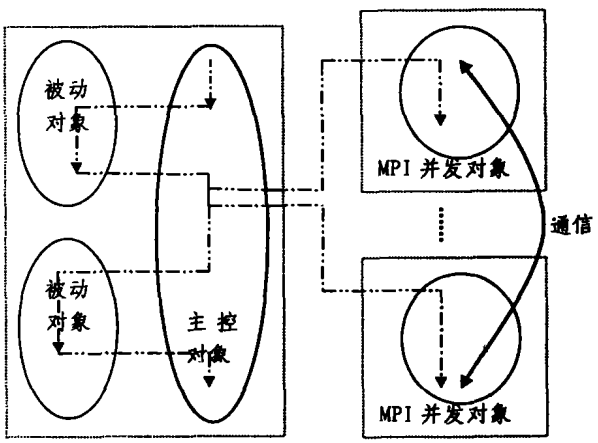

图 1　基于 MPI 并发面向对象模型

在一个并发系统中，计算任务是由若干独立(异步)执行的活动相互合作完成的。为了显式地描述这样的系统，一般要

*)本文研究得到国防预研项目的支持，项目编号为 40106010104。鲁宏伟　副教授，博士，主要研究方向为多媒体系统技术和并行与分布式系统。肖永玲　硕士，主要研究方向为并行与分布式系统。

考虑三个方面的问题：(1)并发执行，描述一个并发系统，首先要考虑采用什么样的结构来表示并发执行的单位，以及如何让它们并发执行。这些并发结构在描述进程的创建方面也各不相同，在有些结构中，进程是静态创建的，进程的个数是固定的；在另一些结构中，进程的创建则是动态的，进程的个数随着系统的执行在变化。(2)并发活动之间的通信。在一个并发系统中，并发执行的活动之间往往需要进行交互，即，通信。在基于进程的并发系统中，进程间的通信可以采用共享变量(shared variables)和消息传递(message passing)两种方式，共享变量通信方式适合于具有共享内存体系结构的计算机中，消息传递则不受共享内存的限制。异步消息传递和同步消息传递都是单向通信，为了方便地描述 client/server 计算模型中的消息传递，一种较高层次的消息传递机制被引进：远程过程调用(remote procedure call 或 RPC)。(3)并发活动的同步。在并发系统中，进行通信的并发活动间往往需要同步。较常见的同步形式是条件同步(condition synchronization)，即，一个活动的继续执行必须要等待一个条件或事件的发生。在基于共享变量的消息传递机制中还存在另一种形式的同步：互斥(mutual exclusion)，即，为了保证用于通信的共享变量不受错误的并发操作的影响，往往要互斥地使用它们。

下面给出本文所提的基于 MPI 的并发面向对象模型的形式化描述：

(1)系统由一个主控对象 MO，一组静态创建的并发对象 CO1，CO2，…，COn，和一组被动对象 PO1，PO2，…，POn 构成；

(2) 每个并发对象 COi 有一组局部变量(v1，v2，…，vr)、一组方法(m1，m2，…，mk)；

(3) 并发对象间采用异步消息传递机制：发送消息 COi. Sj(…)，接收消息 COi. Rj(…)；

(4) 并发对象间还可以进行集合通信操作 COi. Cj(…)；

(5) 方法语句表 m_st_list 由一般语句、异步消息调用语句 COi. Sj(…)、COi. Rj(…)，集合通信调用语句 COi. Cj(…)和局部方法调用 mj(…)构成。

2.1 对象

在基于MPI的并发面向对象模型中，定义三种对象(图1所示)：

(1) 主控对象 MO：即用户程序中的 main()函数。这是一个主动对象，在该模型中，就是由它调用和激活并发对象的。

(2) MPI 并发对象 CO：它是由 PARAMPI 类继承来的对象，是该模型所定义的一个主要部分。正是由该对象才能调用 MPI 函数库的通信函数，它由主控对象激发以后进入 MPI 环境，驻留在不同的处理器上，有自己独立的地址空间，与主控对象并发执行。MPI 对象之间通过消息传递通信。

(3) 被动对象 PO：即一般的 C++对象。在该对象中，被动对象能被主控对象调用，也能被 MPI 并发对象调用。只有在主控对象或并发对象调用其方法时才被激活，它不能产生新的进程。串行和本地执行是它的特点。

在图 1 的模型中，椭圆表示的是对象，虚线框表示的是进程，虚线箭头表示执行线程。实线箭头表示 MPI 并发对象之间的通信。主控对象或 MPI 并发对象调用被动对象就跟面向对象模型中调用普通对象一样，只是调用其中的方法，等到调用完毕再接着执行，并不产生新的进程。主控对象调用 MPI 并发对象后，在不同的机器上产生新的进程，这些进程与主控对象并发执行。并发对象之间通过异步消息传递通信，也可以进行集合通信。

2.2 MPI 对象间的通信

1. 异步消息传递　在所有的并发面向对象模型中，对象间的通信归纳起来可以分为三种：

(1)消息传递机制。消息传递又可分为同步消息传递和异步消息传递。同步消息传递是指消息发送者必需要等到消息接收者接收消息后才能继续执行后面的工作，当没有消息缓冲的时候必须采用这种方式。异步消息传递是指消息发送者不必等待消息接收者就可继续执行，在需要消息结果时再等待。

(2)远程过程调用。发送对象必需要等到接收对象完成发送对象请求的操作返回结果后才能继续执行后面的工作。远程过程调用类似于面向对象模型中的方法调用，不同的是它是在不同机器的不同进程上调用。这种方法没有充分发挥对象间的并发。有的模型采用非阻塞、异步方法请求机制提高并发度，但增加了对方法请求管理的复杂性。

(3)共享变量。并发对象通过共享变量通信，这种方法适合共享内存体系结构的系统。使用条件临界区、管程、计数器、信号量等机制解决同步问题。

为了提高系统并发度和利用 MPI 所提供的通信函数，我们采用异步消息传递机制。在采用异步消息传递机制的并发面向对象模型中，系统的并发是由异步消息发送产生，属于一种细粒度(fine-grained)控制，并发程度较高，由 MPI 控制异步消息传递的复杂行为，提高系统的可靠性和性能。对象发送消息后不必等待接收方接收和处理消息结果就可继续执行，只有在消息接收者需要消息处理结果时再等待。而且异步消息传递提供了阻塞和非阻塞两种方式。

2. 集合通信　利用 MPI 所提供的通信控制器，丰富的集合通信函数，我们也可实现 MPI 并发对象间的集合通信(如图 2 所示)。调用 Barrier 操作，它不发送消息，只是对多个并发对象进行同步。调用 Broadcast 集合操作，一个 MPI 并发对象可以给一个通信控制器中的所有的 MPI 并发对象发送消息；调用 Scatter 操作，一个 MPI 并发对象能够给一个通信控制器中的所有的 MPI 并发对象发送不同的消息；调用 Gather 操作，来自一个通信控制器中的所有的 MPI 并发对象能够将缓冲区中的消息发送给某一个 MPI 并发对象；调用 All gather，一个通信控制器中的所有的 MPI 并发对象能够将缓冲区中的数据发送给所有对象；调用 All to All 操作，一个通信控制器中的所有的 MPI 并发对象能够将缓冲区中的不同数据发送给不同的对象。在该模型中使用集合操作能够大大简化多个并发对象之间的消息传递。

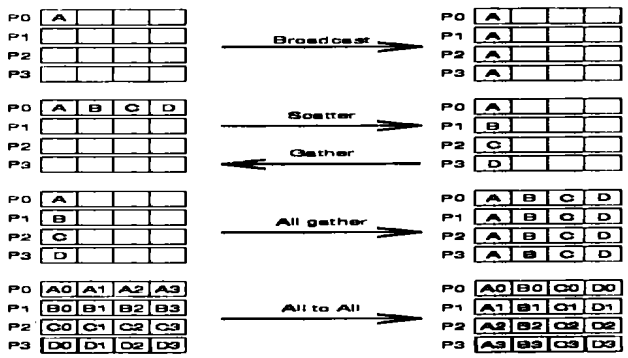

图 2　集合通信操作

2.3 并发对象的同步

在本文所提的模型中，采用集中控制的方式，由主控对象来控制 MPI 并发对象的创建、激发与消亡。对象之间的异步消息传递则由 MPI 支撑系统来控制。

3. 基于 MPI 并发面向对象模型的设计与实现

并发面向对象模型的实现可归纳为如下三种方法[2]：

(1)设计全新的并发面向对象语言；

(2)扩展现存的顺序面向对象语言；

(3)使用现有的顺序面向对象语言，通过外部库提供并发抽象。

早期的大多数系统都可归于第一种方法。如：ABCL/1，POOL、HYBRID、SR[5]等。第二种方法是随着面向对象方法的成熟和顺序面向对象语言的流行而提出的。类库方法，即第三种方法是最近才提出的，国外在这方面已经做了很多工作，提出了一些基于 C++语言的并发类库，如 ACT++[6]。

我们采用类库的方法实现本文的基于 MPI 并发面向对象模型。通过在类库中定义并发类实现在面向对象语言中引进并发。采用这种方法，使得现在已经存在的顺序面向对象开发工具、调试工具和软件仍然可用，而且更重要的是用类库方法具有更好的可扩展性、可移植性和可复用性。并发类库中包含 PARAMPI 类，Message 类等，其中最主要的是 PARAMPI 类。

```
Class PARAMPI
{
public:
    PARAMPI(int &argc,char**&argv) { MPI_Init(argc,argv);
        };//构造函数
    ~PARAMPI(){MPI_Finalize();}//析构函数
    Send(Message*message,int destination );//阻塞发送
    Receive(Message*message,int scource,MPI_Status*status );
        //阻塞接收
    Isend (Message* message, int destination, MPI_Request*
        request);//非阻塞发送
    Ireceive (Message* message, int scource, MPI_Request*
        request);//非阻塞接收
    Barrier();//同步对象
    Broadcast(Message*message,int root);//广播
    Scatter (Message* sendmessage, Message* receivemassage, int
        root);
    Gather (Message* sendmessage, Message* receivemassage, int
        root);
    Allgather(Message*sendmessage,Message*receivemassage);
    All to All(Message*sendmessage,Message*receivemassage);
    int Get_Rank(){MPI_Comm_rank(MPI_COMM_WORLD,
                    &my_rank);
                    Return my_rank;}; //获得并发对象在
                    MPI环境中的秩号
}
```

基于 MPI 的 C++并发类库中 MPI 并发类的实现利用了 MPI 提供的函数执行对象的分配和消息的传递。MPI 并发类的构造函数和析构函数是调用 MPI 函数库的 MPI_Init()和 MPI_Finalize()实现的。MPI 并发类的构造函数使并发类进入 MPI 环境，在 MPI 环境中注册以调用 MPI 函数库中的函数。析构函数使并发类退出 MPI 环境。MPI 并发类之间通信的消息数据的管理和消息的接收和发送也是通过调用 MPI 函数实现的。消息发送 Send(Message* message, int destination)就是调用 MPI 的消息发送函数 MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)，消息接收 Receive(Message* message, int scource)就是调用 MPI 的消息接收函数 MPI_Recv(void * buf,int count,MPI_Datatype datatype,int source,int tag, MPI_Comm comm,MPI_Status* status)。同理，集合通信操作也是调用 MPI 的相应的集合通信函数实现的。

4. 模型的特点

为了更加清楚本文所提出的并发面向对象模型的特点，从以下几个方面说明：

1. MPI 与面向对象语言的集成

面向对象语言与 MPI 的结合大大简化了并发程序的开发。一方面，利用 MPI 提供的函数，使得面向对象语言中的对象能够分布在不同的处理机上，各对象之间的通信也很简单。另一方面，利用面向对象方法，使得 MPI 程序员能够利用面向对象方法的优点(抽象性、封装、多态性、可复用性、继承等)。

2. 提供了并发对象之间的集合通信操作

在一般的并发面向对象模型中，只有点到点的通信。在本文的基于 MPI 的并发面向对象模型利用 MPI 所提供的丰富的通信函数，不仅提供了点到点通信，还提供了如 Broadcast、Scatter、Gather 等多种集合通信操作。

3. 并发控制比较简单

为了让利用该模型开发的并发类库尽可能使用 MPI 的通信函数，类库尽可能简单，容易编程。不同的 MPI 并发对象有不同的控制线索和地址空间，而且 MPI 提供了一个通信控制器使得 MPI 的进程之间的通信很安全，这些使得对并发对象的控制比较简单。

结论 并发技术与面向对象技术的结合是一个新的研究领域。本文基于已经被广泛采用的消息传递标准 MPI 作为底层的并行环境，提出了基于 MPI 的并发面向对象模型，该模型将 MPI 与面向对象集成起来，既利用 MPI 的消息传递与进程分配，又利用面向对象技术中封装、继承、多态、复用等特点，使得所开发的系统具有可扩展性，可复用性。

本文的工作还只是一个初步的研究，进程分配和通信以及简单的并发控制都是利用 MPI 提供的函数，今后我们的工作是采用消息缓冲队列机制提高并行度，更多地利用 MPI 的特点，如通信控制器、自定义数据类型、虚拟拓扑等完善对象间的通信，进一步完善 PARAMPI 类。

参考文献

1 Agha G. Actors: a model of concurrent computation in distributed systems. MIT Press, Cambridge, MA, 1986

2 Karaorman M, Bruno J. Introducing Concurrency to a Sequential Object-Orinted Language. Communication of ACM, 1993, 36(9)

3 Poggi A, Destri G. Using PVM to Develop a Distributed Object-Oriented Language for Heterogeneous Processing. System software, 1998

4 Message Passing Interface Forum. MPI: A message-passing interface standard, June 1995

5 Papathomas M. Concurrency Issues in Object-Oriented Programming Languages

6 Kafura D, Mukherji M, Lavender G. ACT++2.0: A Class Library for Concurrncy Programming in C++ Using Actors