·软件技术与数据库·

【Abstract】This paper proposes a parallel modeling method to improve design decision and reduce time-consuming software development. It describes how to customize a series of models for shared and distributed memory application by using UML extension mechanisms, and how to support real applications by combing these basic models. The method maintains the features of UML such as facilitating understanding, customization, guideline for practical application and extension from sequential modeling to parallel modeling. The usefulness of this method is demonstrated by modeling a real parallel application.

1 概述

随着计算机硬件性价比的不断提高，高性能计算的应用领域在不断扩展，这使得并行系统的规模和复杂性与日俱增。面向高性能计算的并行程序存在着语言繁琐、设计复杂、性能估算不精确等问题，而一种合适、有效的虚拟建模语言能够较好地解决这些问题，达到减少程序开发和维护成本的目的。目前已有许多面向并行程序设计的虚拟建模语言被提出，其中具有代表性的有：针对MPI程序设计的CODE2.0[1]，可视化设计工具Visper[2]，基于主体建模方法面向分布式程序的Starlogo[3]等，文献[4]研究了面向分布式内存的并行程序和网格应用扩展UML建模。它们主要解决面向特定编程环境和特殊的编程模型问题，并且不能与目前的工业标准统一建模语言(Unified Modeling Language, UML)相结合。

本文通过扩展 UML 语义，建立一种通用的并行虚拟建模语言，提供一种并行软件工程的表达工具，满足程序设计不同阶段的需要，指导系统分析员和程序员描绘和理解并行程序。该方法将 UML 中的类图、活动图和协作图组成一个子集，并使用类图来建立并行体系结构的结构模型，使用活动图来描绘计算、通信和同步等操作，使用协作图来描绘各模块协同工作的状态。

使用 UML 给传统的应用开发和维护带来了许多便利，使用 UML 对并行程序进行建模也会带来如下的好处：(1)文档化和可视化现有的应用系统；(2)规范化、可视化、结构化和文档化新的应用程序；(3)性能建模；(4)在开发过程中进行质量控制。

图 1 说明了 UML 所使用的扩展机制。图 1(1)中，用元类 ActionState 对建模元素 action+进行定义。ActionState 用于模拟算法中一种模式的一个步骤。Tags 中用一组标记标明具体的标记号、类型和时间。标记号用于标记建模元素 action+，类型用于表示模块 action+的类型，时间则表示执行完 action+的时间。action+(图 1(2))常用于表示各类标准的单进单出代码块。而标签是用来表示与执行相关的信息。这样的标记值可以根据需要扩展。为了简洁，本文一些例子中的某些属性没有写出。

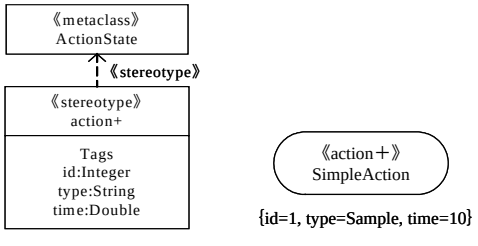

图 1 构造型 action＋的定义

2 处理单元拓扑结构建模

分布式共享内存体系结构可以用 UML 部署图表示。并行程序一般可以映射到一定的虚拟体系结构，本文称之为处理单元拓扑结构。物理体系结构和虚拟体系结构的映射是程

基金项目：国家自然科学基金资助项目(60373008)；教育部科学技术研究基金资助重点项目(106019)

作者简介：胡长军(1963－)，男，教授、博士、博士生导师，主研方向：并行计算与并行编译技术，并行软件工程；丁 良、常晓东，硕士研究生；李建江，副教授、博士

序在目标体系结构上运行时完成的。

文献[4]认为，一个处理单元表示一个进程或者线程。处理单元拓扑可以定义为用拓扑网连接的一组确定的处理单元，这种拓扑与硬件无关，只面向应用程序。这里用一个名为 processingunit 的构造型来表示处理单元。图 2(1)说明构造型 processingunit 的结构，图 2(2)表示了基于 processingunit 的类 Process 的定义，而图 2(3)中用对象 a:Process 来表示类 Process 的一个实例，图 2(4)中用多对象:Process 表示了一组 Process 类的实例。在 MPI 标准中提供了用于建立类似处理单元拓扑的语句，这里以 MPI_GRAPH_CREATE 为例说明处理单元拓扑。

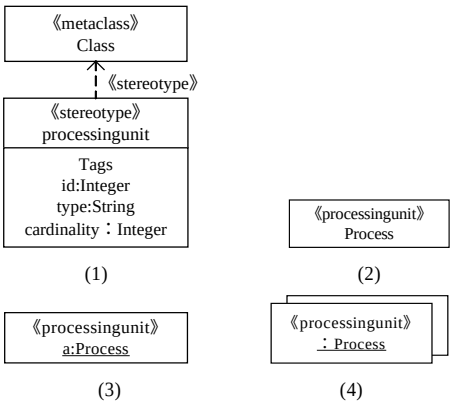

图 2　处理单元建模

图 3(1)是对处理单元的建模，其中，Nnode 表示拓扑图中包含的处理单元数；Index 表示处理单元的度数；Edges 表示这个拓扑图的边数。这里假设处理单元数为 4(Nnode=4)，处理单元度数数组 A=[2,3,4,6]，处理单元边数数组 B=[1,3,0,3,0,2]，它们之间的对应关系见表 1。这个处理单元拓扑可以表示成图 3(2)，其拓扑形式如图 3(3)所示。

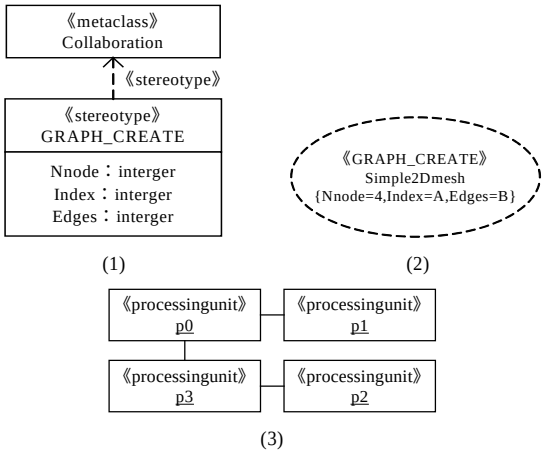

图 3　处理单元拓扑建模

表 1　处理单元映射表

处理单元编号	处理单元度数	相连接的其他处理单元
0	2	1, 3
1	1	0
2	1	3
3	2	0, 2

smp-cluster 是并行计算环境中一种常见的体系结构，本文把 smp-cluster 节点内的进程定义为组处理单元，它本身就是一个处理单元，但可以包含多个处理单元。如图 4(1)所示定义一个组处理单元的构造型，图 4(2)表示了一个包含 2 个处理单元的组处理单元的类，图 4(3)则是一个实例。图 4(4)是一个有 4 个组处理单元，每个组处理单元又是包含 2 个处理单元的一个混合处理单元拓扑，对于组处理单元较多的应用，可以用类似图 3 中的方法表示，图 4(5)和图 4(6)表示了上述实例。

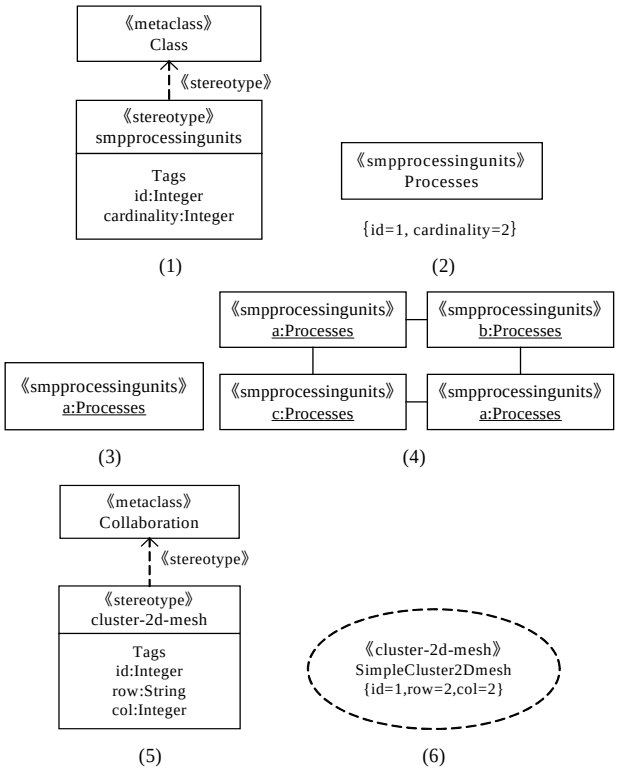

图 4　smp-cluster 处理单元建模

3　应用程序建模

图 5 中分别是串行、共享内存和分布式消息传递并行程序中的一些重要模型。本文主要关注并行模型建模。

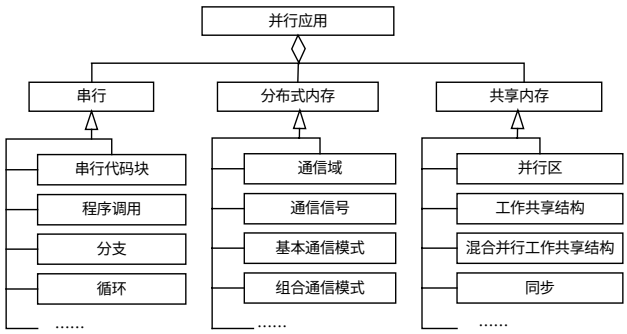

图 5　应用程序建模体系结构

3.1　分布式内存模块建模

在分布式内存体系结构上，大多采用消息传递 MPI 的编程模式来开发并行应用。而消息传递编程模式中最重要的是通信模式建模。最基本的通信模式为点对点通信，其他通信模式都是由处理单元间的点对点通信组合而成的。本文将通信分为源处理单元、目的处理单元、通信信号和通信类型，而对通信模型的建模主要是对通信信号和通信事件的建模。

3.1.1　通信域管理建模和通信信号建模

通信域是 MPI 的重要概念，所有的 MPI 通信都必须发生在一定的通信域内，MPI 标准提供了多个函数进行通信域的创建和管理。这里建立一个新的类表示通信域管理，如图 6(1)

所示，其他具体的通信域管理函数则是它的子类。图 6(2)表示了一个继承自通信域管理的通信域创建的子类，其中，comm 表示目前所在的通信域；group 表示新的通信域中包含的处理单元，这组处理单元是带有拓扑模式的；newcomm 表示新的通信域名。

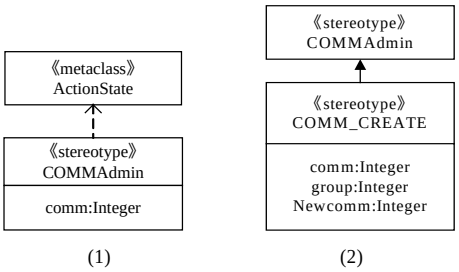

图 6　通信域管理建模

3.1.2　基本通信模型建模

点对点通信(P2P)最典型的通信事件是发送和接收。这样的通信事件分为阻塞通信和非阻塞通信。对于非阻塞发送(non-blocking send)在发送时并不会阻塞发起发送的处理单元，这个处理单元仍会继续向下执行程序。图 7(1)表示了 P2P 的通信信号的 UML 的模型，其中，source 表示发起这个消息的处理单元的 ID 号；destination 表示这个消息的目的处理单元的 ID 号；messageSize 表示这个消息的大小；comDomain 表示这个消息所存在的通信域。图 7(2)表示 P2P 的非阻塞发送通信类型的模型，它是一个基于元类 Transition 的构造型 nbsend，其中，id 表示这个通信模式的唯一标示号。图 7(3)是一个使用 nbsend 向处理单元 b:Process 发送一个 P2P 信号的例子。

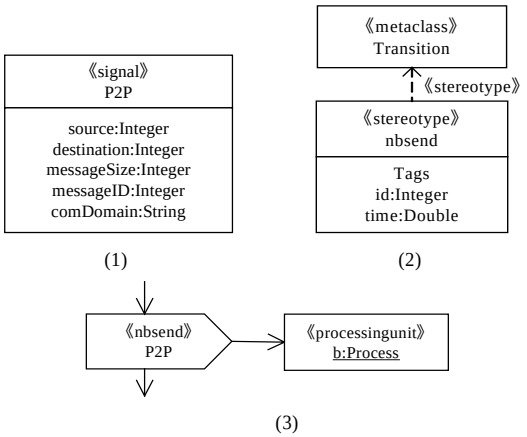

图 7　点对点通信建模

3.1.3　组通信模型建模

由基本通信模式在不同处理单元间组合而成的典型通信称为组通信模式(collective communication)。根据参与通信的处理单元数目和方向可以将组通信模式分为 3 类：一对多通信，多对一通信和多对多通信。组通信可以符合 MPI 标准规定由通信库实现，同时因为并行程序设计的复杂性，也允许程序员根据具体应用问题自行设计和实现新的组通信模式。

通过对基本通信模型、通信域管理和组通信的建模，基本可以表示常见的通信模式。但是在实际应用中，常常需要用这些已有的通信模式设计新的通信。图 8(1)表示的是将本处理单元的 rightbuf 发送到处理单元拓扑中右边的处理单元，而把 leftbuf 发送到左边的处理单元中，并且从左右两边的处理单元接收新的 leftbuf 和 rightbuf。如果这样的通信模式在程序中很典型，而且每个处理单元都做同样的操作，这里将对这种通信模式独立建模。图 8(2)表示的是设计一个新的通信模式 Tran。它的通信信号有以下的属性：与左边处理单元的消息大小 mesSizeLeft，与右边处理单元的消息大小 mesSizeRight，和相应消息的 ID 号，通信域和通信域大小。图 8(3)则是 Tran 在程序中调用时的使用方法。

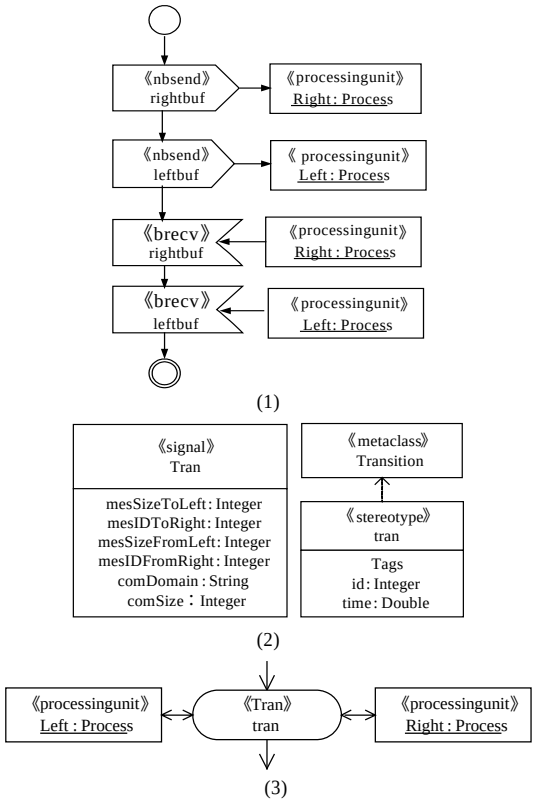

图 8　自定义组通信建模

3.2　共享内存模块的建模

OpenMP 是一个公认的共享存储系统并行编程接口的标准，它由一些语言指导(directives)及库函数组成。它的优点是简单、通用、有利于快速开发并行程序。这里仅以并行区建模为例说明这种建模方法。

如果一段代码区域可以被一组线程并行执行，称之为并行区(parallel region)。为了说明并行区，本文建立了一个新的构造型 parallelregion。因为 OpenMP 指导语句的加入并不影响串行程序的结构，所以新的构造型同样是基于元类 SubactivityState，并通过不同的标记值表示它们的一些重要属性。图 9(1)描绘了基于元类 SubactivityState 的构造型 parallelregion 的定义，OpenMP 的特点使其表现形式与串行代码相同，其中定义的 nrThread 标示了该并行区内处理单元个数。图 9(2)是构造型 parallelregion 的一个例子，右上角的星号(*)表示它是动态并发的。

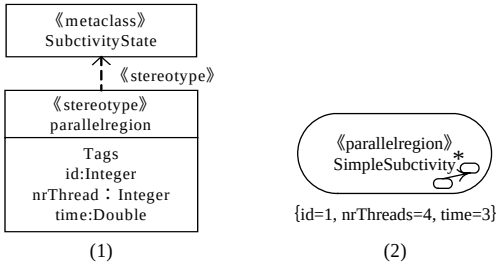

图 9　行代码区建模

4 映射

本文使用泳道(swimlane)来表示模型中动作和子动作的职责，每个泳道一般都对应一个处理单元。图 10 表示将一个单程序多数据流(SPMD)的程序映射到一个处理器单元拓扑上。这里表示了一个存在于名为 Process 的单一泳道中的活动图(图 10(1))。图 10(2)表示了一个多指令多数据的程序。程序存在于 3 个不同的泳道中，它们被映射到一个环状的处理单元拓扑(图 10(3))上。

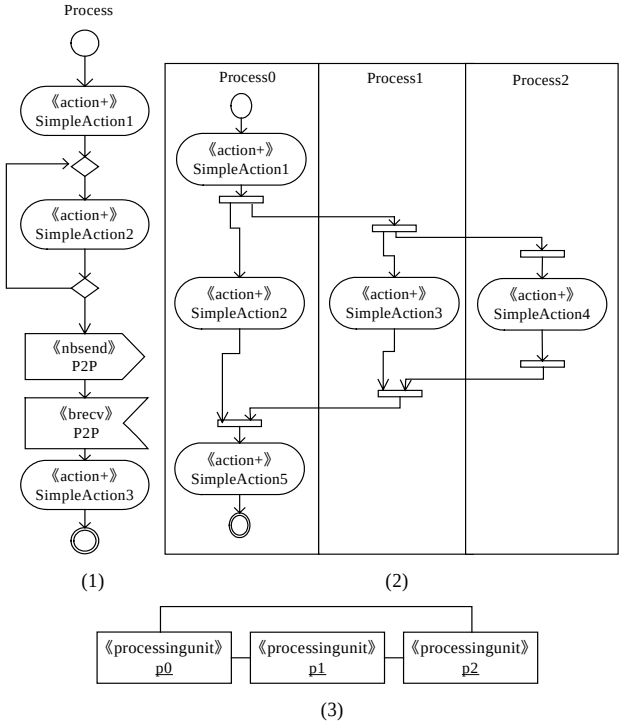

图 10　程序的映射

5 实例

这里使用了 Mpi, OpenMP 进行混合开发。现以一个油藏数模程序为例，说明这种图形化方法在程序设计各个阶段的应用。在程序设计的初始阶段，对程序进行粗粒度的描述。

图 11(1)代表了整个程序，它由 TRANS, SOLMAT 和 QRATE 3 个子函数模块组成。图 11(2)着重对 solmat 模块的进行说明，它调用了 BUNDRY, WELL 和 JCG 3 个子函数，并有一个并行区 CALCULATE PRESSURE 和一个工作共享结构。图 11(3)表示了实验的面对的处理单元的拓扑结构。图 11(4)细化了 JCG 模块。该模块使用雅可比共扼梯度法解油藏数模中的压力方程，它由 5 部分组成。其中，Data Partition 负责将数据按照并行方案分布到各处理单元上；Communicate for Compute 用于完成组织本次迭代所需数据的矩阵；COMPUTE NEW ITERANT 用于计算本次迭代；COMPUTE FOR STOPPING TEST 用于计算迭代的敛散性；Data Gather 在计算完成前收回最终的解矩阵。图 11(5)是在迭代开始前准备数据的 Communicate for Compute 阶段，该函数分别向左右处理单元分发一个 bufer，为了提高通信效率，本文采用了非阻塞的发送；由于它需要分别从左右处理单元接收一个 bufer 以完成自己负责的迭代部分，因此为了保证接收的正确，采用了阻塞的接收。图 11(6)说明了迭代主体 COMPUTE NEW ITERANT 是如何运行的。这是一个三维矩阵，它的迭代方程在 *x, y, z* 方向上是独立的，最后则需要将 3 个方向上的计算结果与原值求和得到迭代后的解。因为这些循环迭代是并行执行的，所以最后的求值是个混合并行工作共享结构。

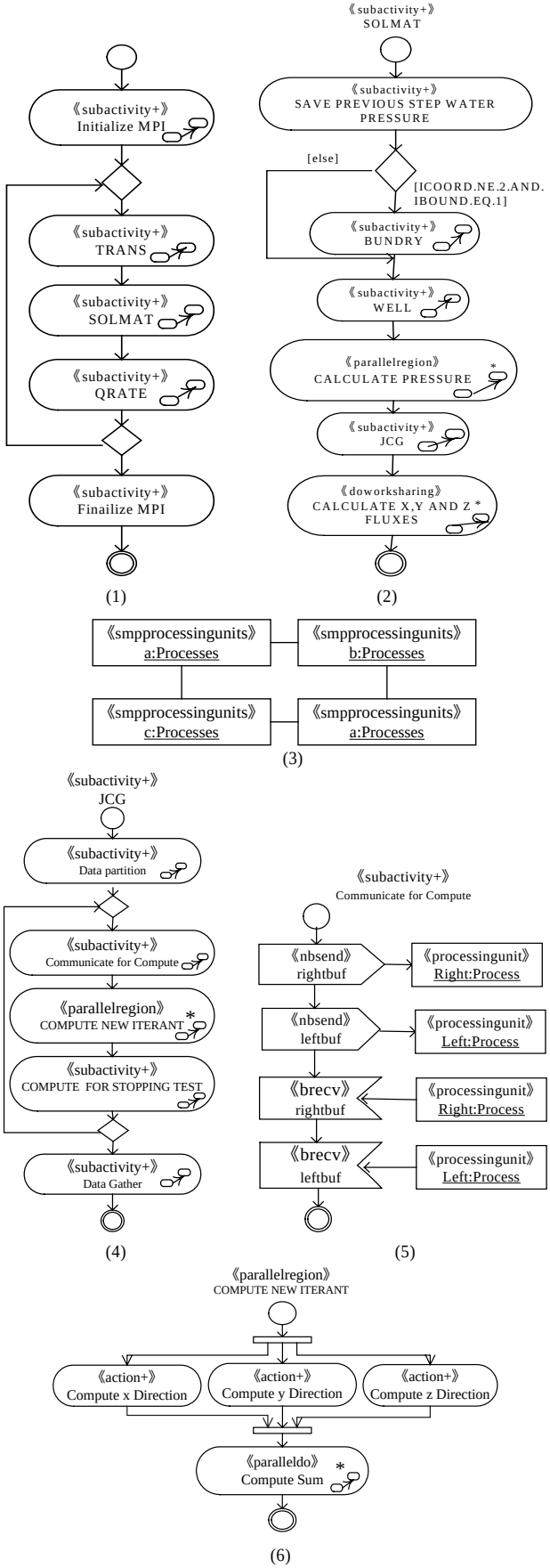

图 11　油藏数模并行化建模

（下转第 93 页）