信息聚焦下的任务分解方法①

Task Decomposition Technique in Information Focus

林金芳　张天刚　(江南计算技术研究所 江苏 无锡 214083)

摘　要： 信息集成虽然消除了信息源之间的信息异构，但仍然无法针对信息需求实现高效、准确的信息聚焦。为此，在给出任务定义的基础上，提出了一种基于任务流程的任务分解算法，把任务分解为具有不同优先级的子任务，找出子任务间数据依赖和控制依赖关系，为有效实现信息聚焦提供依据。

关键词： 信息聚焦　任务定义　任务分解　控制依赖　数据依赖

1　引言

长期以来，数据集成一直是信息管理领域研究的重点。数据集成的研究主要是为了消除不同数据源之间的数据异构，克服数据共享的“基本障碍”，将互相关联的分布式异构数据源集成到一起，使用户能够以透明的方式访问这些数据源 。国内外的研究人员在该领域做了大量的工作，开发出了一系列的应用系统：典型代表有 TSIMMIS 系统、MOMIS 系统[1]，基本实现了异构信息共享。

虽然信息共享消除了不同信息源之间的信息异构，克服了信息应用的“基本障碍”，但用户所面临的依然是浩如烟海的“可得”信息(available information)，难于实现“准确的信息在适当的时间为需要的人所利用”[2]。而提高组织决策质量和效率，对环境和问题做出更为快速准确的评估和判断、采取更佳行动，需要及时高效、准确满足用户及信息应用的能用好用管用的“有用”信息(useful information)。在“可得”信息与“有用”信息之间存在一条巨大的“鸿沟”，制约了信息应用的广度和深度。其存在的根本原因在于信息、信息需求与衔接两者的信息处理技术之间的脱节。如果说信息共享提供了对信息进行统一处理的平台，那么如何有效地描述用户和信息应用的信息需求，进而基于信息需求采用高效的信息处理技术在广泛的信息空间进行“聚焦”，将“准确的信息在适当的时间”提供给用户是信息应用的关键所在，这就是我们所说的信息聚焦。

由此可见，信息集成仍然无法针对信息需求实现高效、准确的信息聚焦。因此我们就需要对用户和信息应用的信息需求进行描述，在这里我们将用户和信息应用规范成一个任务，那么这个问题也演变成对任务信息需求的描述。但是在实际应用中，任务多数时候是一个复杂任务，要依据复杂任务实现信息聚焦是一个复杂问题。因此在大型网络应用中往往需要对任务进行分解，并依据任务分解的结果来实现信息的聚焦。在这里我们只讨论如何对任务进行分解。

任务的分解问题在相关领域已有很多的探索，特别是在并行计算领域，从对串行程序的并行划分[3]到对程序切片[4]的研究，逐步深入，但是对于大型网络应用中的任务，如果采用程序级的划分方法，那么会产生为数众多的子代码，这会使信息聚焦过程中的通信变得非常烦杂，即使对划分后的子代码再进行组合以增加粒度，也会破坏信息聚焦的完整性和划分的准确性。其它的划分方法有均匀划分技术、方根划分技术、对数划分技术和功能划分技术等，这些方法对于数据并行机制是有效的，而对于数据不规则的任务，例如信息聚焦下的任务，就需要设计出适合其任务特点的分解方法。针对信息聚焦快速、高效、准确等特点，本文将从适合并行执行的角度出发，通过对聚焦任务进行分析，设计适合聚焦任务特点的任务分解方法。

2　任务的形式化描述

任务分解的主要功能是将提交的任务分解成多个具有尽可能高并行度的子任务，并确定子任务间的依赖关系[5]。在任务分解的过程中，往往要先建立子任务之间的纵向关系以支持自顶向下(top-down)[6]的流程设计，这样能够更好地反映出任务的管理层次。

2.1 任务定义

在任务定义时，我们给出以下一些定义：

① 收稿时间:2008-08-16

定义 1.一个任务可由多个子任务组成，关系 Sub(A,x)，若 x 不为空，则 x 是 A 的子任务集。

定义 2.关系 Seq(A_1,A_2,A_3…)是一个偏序关系，表示子任务 A_1,A_2,A_3…按时间顺序执行；关系 Para(A_1,A_2)，表示子任务 A_1,A_2 并行执行，关系 Sele(A_1,A_2)表示在整个任务执行中子任务 A_1 与 A_2 之中只有一个可以被执行；关系 Circ(A_1,y),表示子任务 A_1 循环执行，直到满足约束条件 y。

定义 3.原子任务是没有子任务的子任务，是直接能够分配给单个执行者执行完成子任务，是任务定义的最小单元，若 A 为原子任务，则 Sub(A,x)—>x = ⊙。

规则 1Seq(A_0,Para(A_1,A_2),A_3)表示在子任务 A0 执行后，流程中的控制流一分为二(AND-Split)，子任务 A_1,A_2 并行执行，当 A_1,A_2 都完成后，流程中的控制流聚集到一条(AND-Join)，执行下一个子任务 A_4。Seq(A_0,Sele(A_1,A_2),A_3)表示在子任务 A_0 执行后，流程中的控制流依据选择条件执行子任务 A_1 或 A_2(OR-Split)，当 A_1 或 A_2 完成后(OR-Join)，控制流执行下一个子任务 A_4。

定义 1 定义了任务是由一组子任务组成的；定义 3 定义了分解的粒度。我们将整个聚焦任务看成一个大任务，而通过自顶向下进行任务划分，划分出许多层次化的子任务，直到分到原子任务。而子任务间的层次关系和流程关系则通过定义 2 和规则 1 来刻画，这样的纵向层次划分不仅使得任务更适合实际的应用，而且任务的执行最终也能反映在原子任务执行上。

同时我们设计一个任务定义知识库，该知识库支持两种方式的任务定义：基于任务定义知识库和人工定义知识库。基于任务定义知识库主要依据知识库中的已有任务定义例子，因此，需要将已经完成划分的任务及时保存，并形成知识源库。人工定义是采用交互模式，由设计人员对任务提出求解方案，通过设计人员间的不断对话，最终形成任务划分解决方案，并将任务定义结果入库，形成新的知识源。

2.2 形式化描述

在任务定义的基础上，我们给出任务的形式化描述：

1)任务：一个信息聚焦任务由多个子任务组成，用 T 表示。

2)控制依赖(control dependence)：对于控制依赖的定义，引用 Ferrante[7]的定义，一个控制流程图是一个有向图，这个有向图中只有一个入口和一个出口，图中每一个节点至多有两个后继，两个后继的特性是 True 或 False，并且对于控制流图中的任一个节点，都存在一条从入口到此节点的路和一条从此节点到出口的路。对于控制流程图中的节点 X 和 Y,如果 Y 控制依赖于 X,则 X 必然有两个出口，一个出口导致 Y 被执行，一个出口导致 Y 不被执行。控制依赖用数组 cDep[][]表示。

3)数据依赖(data dependence)：数据依赖是指在本次任务执行过程中，某子任务需要一组其他子任务的执行结果或传递的变量，这组子任务称为此子任务的数据依赖。数据依赖用 dataDep[][]表示。

4)原子任务：原子任务用 S 表示。

5)子任务：子任务用(T,S,V,O,R)表示，T(task)代表子任务所属的总任务，S(subtask)代表原子任务集，V(variable)表示在这个子任务中用到的其他子任务的变量集。O(organize)代表能够完成该子任务的组织角色,R(resource)代表完成该子任务的资源描述。

2.3 任务控制流程图

根据任务定义，容易生成子任务控制流程图，把每个子任务表示为图中的一个节点。因为在图中很容易找出子任务之间的控制依赖关系，所以称之为任务控制流程图。为了便于任务分解算法的执行，先对控制流程图进行形式化处理，图 1 中给出了任务 T 的控制流程图，任务 T 是击毁敌空目标。根据任务定义任务 T 由 3 个原子任务组成：S_1 表示获取敌方导弹来袭信息；S_2 表示发射拦截导弹；S_3 表示获取导弹拦截效果信息。根据信息聚焦下任务的特点，子任务之间的关系可以归结为数据依赖关系和控制依赖关系。控制流程图包含节点之间控制依赖关系，也包含有数据依赖关系，但控制流程图所表示的各节点的顺序关系存在冗余，有些节点可以提前执行，任务分解的目的之一就是要找出能够与其祖先节点并行执行的节点。由于任务的规模较小，可以根据任务流程从直观上判断任务的依赖关系，节点 2 中的 S_2 需要节点 1 中的 S_1 的执行结果，也就是说节点 1 是节点 2 的数据依赖，节点 2 必须在节点 1 之后执行，同样根据控制依赖的定义，节点 5 控制依赖于节点 3；实际中面临的任务是复杂而又庞大的，后面会介绍如何通过算法产生数据依赖，并根据控制依赖和数据依赖把大规模的聚焦任务分解为小规模的并行子任务。

3 任务分解算法

实际用中任务可能包括多个控制流程图，这里考虑如何对一个控制流程图进行分解，找出子任务间的数据依赖和控制依赖，并找出可以并行执行的子任务，

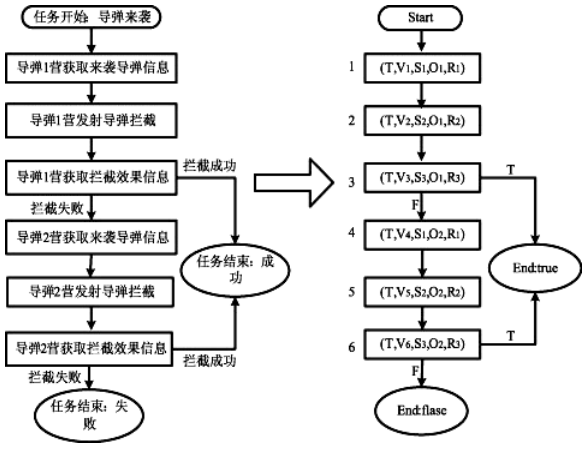

图 1　任务控制流程图的形式化描述

以避免冗余计算，优化任务执行效率，从而快速高效地实现信息聚焦。

3.1 算法描述

任务控制流程图是一个有向图，可以看成是一种层次结构，对它的广度优先遍历实际上是按层次遍历。首先对有向图广度优先遍历求出子任务的控制依赖(算法 1)，在遍历的同时把图以层次的顺序存储在双向队列中，以便于求出子任务的数据依赖(算法 2)，然后根据子任务的控制依赖和数据依赖分解任务，产生子任务的优先级顺序(算法 3)。依赖关系图和优先级次序图是实现任务有效调度的依据。

算法 1 遍历任务控制流程图，产生子任务的控制依赖。

算法思想：对有向图广度优先遍历，判断每条弧的属性，如果为 1(true)或 0(即不是 true 也是不是 false)，则弧头节点控制依赖于弧尾节点，如果为 2(false)则表示无控制依赖。

输入：DG(directed graph)//输入为任务控制流程图。

输出：cDep[][]，DQ//cDep 为二维数组，存放从第 1 个节点到第 n 个节点的控制依赖的节点序号。DQ 是以层次顺序存储有向图的双向队列。

算法描述：从任务流程图的根(root)出发广度优先遍历图，在层次遍历的过程中把图中的顶点按深度顺序放入双向队列 DQ 中。附设队列 Q，其中存储已被访问路径为 1,2,…的顶点，其中 Q 最后为空。

(1)初始化双向队列 DQ 和队列 Q，图的起始节点 root 加入队列 Q。

(2)如果队列 Q 不为空，队头元素 v 出队，v 和其深度值 d 加入双向队列 DQ 中。判断 v 的入弧的属性是否为“2”，若不为 2，则把 v 的前驱加入 cDep[v]，否则 v 的控制依赖 cDep[v]为 NULL。

(3)有向图中节点 v 的所有邻节点依次加入队列 Q 中。

(4)如果队列 Q 为空，则算法结束，否则转第(2)步。

算法 2 对双向队列按层次逆向遍历，产生子任务的数据依赖。

算法思想：对算法 1 产生的双向队列从队尾元素开始遍历即可实现对有向图的逆向按层次遍历，对每个节点检查其祖先节点是否为其数据依赖，从而产生各个节点的数据依赖。

输入：DQ。

输出：dataDep[][]//输出为数据依赖和子任务依赖图。

算法描述：从 DQ 的队尾节点开始对双向队列 DQ 中的节点按顺序访问，对每一个节点都要根据它的变量集 V 来判断它是否需要祖先节点提供变量，如果需要，则它的祖先节点就是它的数据依赖，反之跳过。附设两个指针 P 和 F 辅助访问双向队列 DQ，P 用来指向被遍历的节点，F 用来指向被遍历节点的父节点。

(1)如果双向队列 DQ 不为空，DQ 的队尾节点 P 出队，F=P 的第一个祖先；如果双向队列 DQ 为空，则算法结束。

(2)根据 P 和 F 的变量集，判断 P 是否数据依赖于 F，如果是，F 被加入 P 的数据依赖集；如果不是则转第(4)步。

(3)P 的每一个祖先 F 若都被访问了，则转第(1)步。

(4 F＝P 的下一个祖先；转第(2)步。

算法 3 遍历有向图中节点，对每一节点根据其控制依赖和数据依赖设定其被执行的优先级。

算法思想：算法 1 和算法 2 已给出了有向图中节点的控制依赖和数据依赖，图中控制依赖和数据依赖为空的节点即可与它的祖先并行执行，对于控制依赖和数据依赖不为空的节点，它必须在其控制依赖和数据依赖节点执行完后再执行，但它可以和一些与它无依赖关系的节点并行执行，所以用设定优先级的方法来梳理节点的关系。

输入：DG,cDep,dataDep//输入有向图、控制依赖和数据依赖。

输出：Pri[]//输出为各个节点的优先级。

算法描述：判断有向图 DG 中的每一节点，如果某些节点的控制依赖和数据依赖都为空，则去掉这些节点和所有以此节点为弧头的弧，这些已访问节点组成一个优先级为 1 的集合，产生一个弱连通的图 DG2，对 DG2 中的节点进行类似的处理，首先找出所有控制依赖集和数据依赖集是已访问节点集合的子集的节点，设定它们的优先级，然后去掉这些节点和所有以此节点为弧头的弧，以此类推，直到对所有的节点都设定了优先级。优先级值越小，则优先级越高。

(1)Prior=1，所有节点标志为未访问。

(2)找出未访问节点中所有控制依赖集和数据依赖集为已访问节点的子集的节点 v，设定 v 的优先级：Pri[v]=Prior，把节点 v 标志为已访问。Prior=Prior+1。

(3)如果所有节点都被访问则算法结束，否则转第(2)步。

3.2 任务分解示例

图 2 给出了对图 1 中定义的任务 T 的分解过程，图 2(a)给出了任务 T 的任务控制流程图的简化图，每个子任务是有向图中的一个节点。图 2(b)中的虚线弧表示控制依赖，实线弧表示数据依赖。

通过算法 1 可以得到这些节点的控制依赖，通过算法 2 可以得到这些节点的数据依赖。得到子任务的控制依赖和数据依赖(见表 1)之后，利用算法 3 可以得出各个子任务的优先级(图 2(c))。从图 2(c)可以看出，子任务 4 可以和子任务 1 或子任务 2 或子任务 3 并行执行，子任务按照优先级 1,2,3,4,5 的次序执行，这些关系是制定任务调度规划的依据。

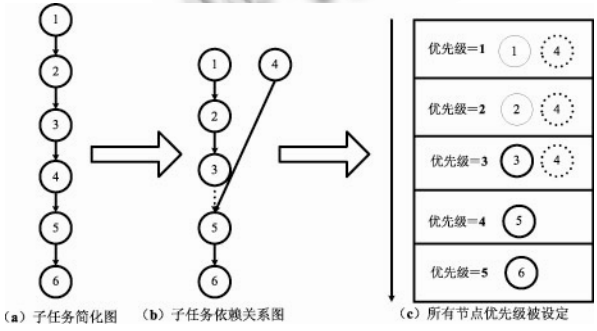

图 2　任务分解示例

表 1　子任务的控制依赖和数据依赖

子任务	子任务控制依赖	子任务数据依赖
1(T,V1,S1,O1)	Null	Null
2(T,V2,S2,O1)	Null	1
3(T,V3,S3,O1)	Null	2
4(T,V4,S1,O2)	Null	Null
5(T,V5,S2,O2)	3	4
6(T,V6,S3,O2)	Null	5

4　结论

本文从适合并行执行的角度出发，提出了适合信息聚焦下任务特点的任务分解方法。确定了子任务间的控制依赖和数据依赖关系，并明确了子任务执行的优先级次序。文章的进一步研究工作就是以任务的依赖关系和优先级次序为基础，如何有效地实现基于任务的信息聚焦。

参考文献

1 Bergamaschi S, Castano S, Vincini M. Semantic integration of heterogeneous information sources.Data & Knowledge Engineering, 2001,36 (3): 215－249.

2 Lenzerini M. Data Integration: A Theoretical Perspective. Proceedings of the ACM Symposium on Principles of Database Systems, 2002:233－246.

3 Ho T, Medard M, Koetter R. An information theoretic view of network management.IEEE INFOCOM, 2003.

4 Morel B, Alexander PA. slicing approach for parallel component adaptation. 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2003: 108－114.

5 陈勇,戴先中.基于移动机器人的可重构多智能体系统平台的构建.东南大学学报,2003,9.

6 Hu JM, Zhang SS. An agile workflow system support vittual enterprise. Computer Research and Develpoment, 1999,36(12):1517－1523.

7 Singh A, Pande S. Compiler optimizations for java aglets in distributed data intensive applications. Proc of 17th ACM Symposium on Applied Computing (Agents, Interactions, Mobility and SystemTrack), 2002: 87－92.