计算机科学1998Vol.25 №.5

支持协同工作的超媒体版本模型

A Hypermadia Version Model Supporting Cooperative Work

吴海英 文栋辉 赵振西

(中国科技大学计算机系 合肥230027)

摘 要:本文综合面向任务和面向状态的概念,提出了一个超媒体版本模型,使系统具有跟踪协同变化的能力,始终可以保持一致的状态,极大地降低了用户的认知负载。同时,系统能很好地支持协同工作,并能在各种工作模式间平滑切换。

关键字:超媒体,版本模型,协同工作,面向任务,面向状态

自从Halasz[1988]提出将版本控制作为超媒体系统的一个研究方向以来,超媒体系统的版本控制已引起越来越多的重视。超媒体系统已被用于各种开发性的任务,在这些任务中用户需要版本支持来进行诸如写作和思想组织、软件构件管理等活动。然而将版本控制并入超媒体系统之后,用户在版本产生时遇到了认知负载问题,在版本选择时则遇到了迷路问题。而且,过多的版本产生也影响了系统的实现效率。虽然对于多用户并发控制的有效解决不可避免牵涉到版本问题,但在协同工作环境下,现有的版本模型则显得不够有效。究其主要原因,版本化在这些应用中不可被剪裁,它是系统强加给用户的,这不可避免地增加了用户的认知负载,降低了系统的灵活性。

针对以上问题,我们综合面向任务和面向状态的概念,提出了一个超媒体版本模型,并给出了具体的版本控制机制,目的是尽量降低用户的认知负载并且提供一定的灵活性,以便可应用于各种超媒体应用。

1. 面向状态的超媒体版本模型

当前绝大多数版本模型多是面向状态的,如Hyperform[1]、HB3[2]等,它们将每个对象的所有版本保存在一个版本集合中,而由其它对象所构成的复杂对象必须通过精确地选定每个构件对象的版本来定义。我们以前给出的超媒体版本模型[3]即是面向状态的。

但是当我们将上述模型应用于超媒体文档写作活动时,遇到了一个难以解决的问题,即系统不能完整地考虑许多超媒体对象的协同变化。如在写作过程中,对于某一内容的修改可能会涉及到大量的复合结点、原子结点和链,这在面向状态的版本模型中是难以实现的。虽然复合结点的复合机制[4]可以部分解决这一问题,但在许多情况下,需要协同变化的对象之间并无具体的链结关系。同样,在面向状态的模型中,大量不一致的超媒体网络中间状态被保存,例如假定复合结点C是由两个原子结点A_1和A_2所组成,A_1和A_2的修改存在着某种协同关系,则当用户对A_1修改产生新的版本之后(A_2还没有改变),C产生了一个新的版本(版本繁殖),而C的此版本在语义上可能是错误的,即超媒体文档发展历史过程中,大量的不一致的状态被系统保存,既影响了系统的实现效率又增加了用户的认知负载。为此我们提出了结合面向任务的版本概念来进行超媒体系统版本控制。

2. 面向任务的超媒体版本模型KDVer

面向任务的方法最主要的一个思想就是为维护那些需要协同变化的版本之间的关系提供系统支持,这样系统能够排除不期望的构件版本之间的组合,始终可以保持一个一致的状态,那些不一致的中间状态对于生成的结果来说是不可见的。

吴海英,文栋辉 硕士研究生,主要研究兴趣为开放式超媒体系统、面向对象数据库。赵振西 教授,博士生导师,主要研究领域为数据库系统,面向对象程序设计和多媒体系统。

面向任务的概念最初是在一个软件开发环境PIE[5]中提出来的，它的主要目的是将一个协同的几个部件组合进一个可以识别的单元——层次，一旦一个层次被产生，所有在此层次的修改被自动记录在一个新的软件系统的版本中。后来，面向任务的版本概念也被应用于数据库系统，如O_2。第一个将此概念应用于超文本模型的可能是HyperPro[6]，它为每个复合结点保存了一个选择准则，用来动态地选择每个原子结点的版本，但它的选择准则主要是基于时间的，这对于一些复杂的系统，如超媒体写作，不能有效地进行模型化。第一个真正地利用面向任务版本概念的是超文本写作工具SEPIA[3]，它将工作过程分为四个空间：内容、计划、讨论和修辞，每一个空间有一特定的空间浏览器与之关联，但该系统的模型显然是针对具体的应用的，缺乏可剪裁性，同时，对于协同工作中的协同模式支持的粒度不够。

将面向任务的版本模型应用到我们的超媒体系统KDVer中，用户对超媒体网络的改变被视为完成一个任务，这些任务能够引导自动的、一致性的版本产生，也即用任务的概念来维护用户在完成一项工作中所产生的和使用的各种对象的版本。针对不同的应用，任务可能代表不同的概念，如在超媒体文档写作应用中，一个任务T可能代表某一章节内容的生成；而在CASE中，T可能代表对某一模块的编译（即目标代码版本的产生），因此任务版本模型可以被应用于多种超媒体应用中。为了增强任务对各种应用模型的描述能力，我们定义一个任务可以有多个子任务，而在这些子任务之间存在着一定的时序关系：前趋或后继。这种时序链的引入，可以更加有效地模型化各种应用。如图1中对于某个超媒体文档的写作任务T_1是由三个子任务所构成，而在这些子任务中，T_{13}只有当T_{11}和T_{12}均完成之后才可以被开始调度执行。

对于任务，可以有三种状态：休眠、活跃及完成。任务在产生时都处于休眠状态，若一个任务有多个前趋任务，则只有当所有前趋任务均处于完成状态时，此任务才可由休眠变为活跃，也即此时用户可以对该任务所表示的超媒体子网进行修改。而若一个任务无前趋任务，它为某一父任务的最左的叶结点，则只有当父任务变为活跃时，该任务才变为活跃。一个已完成的任务，其内部所包含的各种对象的版本可以被其父任务可见，而一个处于活跃状态没有完成的任务只能被其子任务所见（这在实现中由类层次关系确定，见图2）。若一个任务有多个前趋任务，则此任务的初始状态包含所有这些前趋任务的对象版本。值得注意的是，若某一个对象在这些前趋任务中存在着不同的版本，则开始执行前将这些不同的版本进行合并，产生单一的新版本包含进该任务中，因为任务中某个特定的对象其版本是唯一的。若一个任务有一个或多个父任务又无前趋任务，则该任务从它的父任务中继承对象版本。由此可见，任务的概念只是保存了两个状态：开始执行时和完成时，而这两个状态相对于该任务而言均为一致性的状态，那些需要协同变化的对象在任务执行过程中改变，不一致的状态处于任务执行过程中而不被保存。

对于任务中一个对象版本V的修改，得分几种情况：如果V是在T执行过程中产生的，则将V加入到T中，以后的修改在V中进行；如果V是T开始执行时的版本，则产生一个新的版本V'，并将V'代替任务T中的V，以后的修改在V'中进行。需要指出的是，以上过程中版本的合并、新版本的产生以及任务中对具体哪个版本的修改都是由面向任务的概念自发而完成的，用户无需知道版本的概念，该模型提供了一个缺省的版本产生机制和跟踪协同变化的机制，用户唯一要做的是划分任务层次和规定每个任务所涉及到的应用对象的范围，因为这在不同的应用中是不同的。如在超媒体协同中，用户只需指定每个任务所牵涉到的超媒体子网（有可能是不相连的）。可见，该模型极大地降低了用户的认知负载，又可适用于多种应用，而这些应用无需被扩展以支持版本功能。

为了增强该模型的灵活性，我们将面向状态的概念与面向任务的概念结合起来，将所有在任务执行中产生的新版本记录在版本组结点中，允许用户通过面向状态的操作对该版本进行检索。同样，在任务执行过程中，应用可以通过两种面向状态的版本操作（包含和排除）对一具体的超媒体对象版本进行选择。若用户通过排除命令将V从T中排除，则从T中移去V。若一个版本V被用户包含进任务T，则将V加入到T中。但以后若对V进行修改的话，则系统自动产生一个新的版本V'，以V'代替T中的V，所有的修改均在V'上进行，这保持了任务状态的一致性。同样，在任务执行过程中，用户可以显式地发出一冻结命令，保存当时任务T中的状态（如果用户觉得必要），将该状态提交给任务T的父任务。

现在，用户若想回到超媒体文档发展历史过程中的某一时刻的状态，可以有两种途径：一是通过面

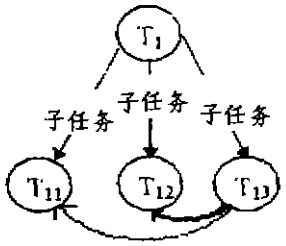

图1 一个任务的实例

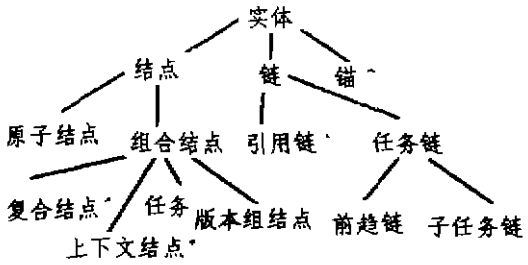

图2 超媒体版本模型的类层次

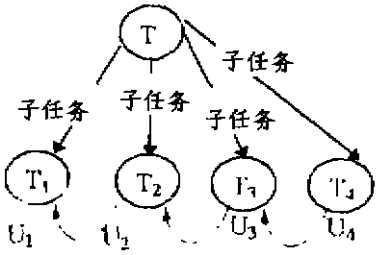

图3 个人工作模式

向状态的操作在版本组结点中精确地指定该时刻所包含的超媒体对象以及所有这些对象的具体版本，这显然是容易出错的，因为对于每个复合结点还需具体指定它所包含对象的精确版本；二是借助于任务的概念，即某个任务 T 的完成时刻或开始时刻的状态，只需指定某个具体的任务，因为任务的本身是由用户定义的，而由系统通过保存于任务中的对象版本自动生成当前时刻的一致性状态，这极大地降低了用户的认知负载。同样，KDVer 模型也减少了过多不必要的中间版本，因为此时的版本繁殖是针对一个任务而言，而不是针对整个超媒体网络。例如某一个复合结点 C 没有被包含进任务 T，若 C 中内容 A_1 在 T 中产生了新的版本，此时 C 仍旧保持原来的版本（针对 T），这在面向状态的模型中必然要生成一新的版本。

3. KDVer 模型对协同工作的支持

超媒体技术已经被应用于多种应用领域，如 CASE、CAD、协同写作等，这些应用中一般都需要多个用户对超媒体网络进行协同存取，因为传统的数据库技术并不能有效地支持协同，利用超媒体技术进行协同工作[8]已受到越来越多的重视。这样的系统一般分为三类：实时系统、非实时系统和混合系统，超媒体写作系统就属于混合系统。下面我们就讨论一下 KDVer 版本模型对协同工作的支持。一般的协同系统显然能支持多种协同模式，但大多数这些系统在同一时刻只能支持唯一的协作模式并且用户在不同的协同模式之间的转换非常不易。

协同系统包含四种协同模式：个人工作模式、分离工作模式、松耦合工作模式及紧耦合工作模式。个人工作模式指在任一时刻系统内只有一个用户工作于信息库中，这也即一般的单用户系统。利用面向任务的版本模型，我们可以用图3所示进行模拟。此图中，任务 T 由多个子任务所组成，而子任务间存在着严格的线性关系（由前趋链所决定），用户 $U_1U_2U_3U_4$ 可能相同或者不同，但这保证了同一时刻只能有一个用户工作于该信息库中并且看到该信息库的内容，因为一个任务可以被执行当且仅当其所有的前趋任务均完成之后才可以执行。

分离工作模式允许多个用户在同一时刻并行工作于同一信息库中，但每个人分别存取各自的超媒体子网，这些子网之间是没有交集的，即这时多个协同用户可以并行工作而不受别人的影响，我们可以用图4对这种模式进行模拟。图4中的用户 $U_1U_2U_3$ 工作于分离工作模式中，在该情况下，任务 T 对它的子任务进行控制，即不允许任意两个子任务存在交集，也即不允许任意两个用户存取同一超媒体网络对象，而由于各个子任务之间无前趋关系，故可以并行进行。当它们所有的任务完成之后，由 T_4 对每个用户所存取的超媒体子网进行合并。当然，也只有当这三个子任务均完成之后，任务 T 才真正地完成，这提供了一种并行工作模式。

松耦合工作模式允许不同的用户对同一结点进行编辑，但每个人的工作均保持一定的独立性，系统提供一定的机制让用户相互了解各自的工作进展以减少写冲突。这一模式与图4类似，但是该任务 T 的所有子任务之间可以有相同的超媒体对象，如前所述，每个任务在执行前均继承了 T 的所有对象版本，但每当一个子任务对一对象 O 进行修改时，均

是在 O 的一个新的版本上进行，即假定子任务 $T_1T_2T_3$ 都包含了对象 O，它们的编辑在互不相同的版本 $V_iV_jV_k$ 之上进行，这体现了松耦合模式的本质，而在任务 T_4 中则需对这几个版本进行合并以生成一个唯一的一致性版本。当然每个用户在工作过程中，可以通过查询版本组结点中该对象的其余版本来了解别人的工作进展以减少写冲突。

紧耦合模式即 WYSIWIS 的模式，保证了各个协同者所见的一致性。如图5所示，图中 $U_1U_2U_3$ 工作于紧耦合模式。由于 T_4 继承了 $T_1T_2T_3$ 的所有对象版本，而 $T_1T_2T_3$ 又继承了 T 的对象版本，所以 T_4 包含了所有 T 的对象版本，当用户对 T 中的对象进行修改时，则产生一个新的版本 V'，以后任一用户对该对象的编辑均在 V' 中进行，因为任务中只包含一个对象的唯一版本，这保证了不同用户所见的一致性。

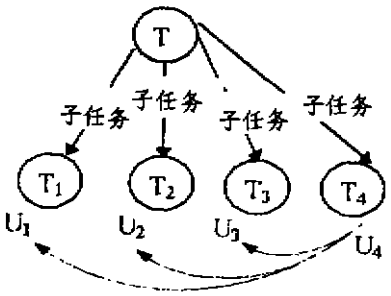

图4 分离工作模式

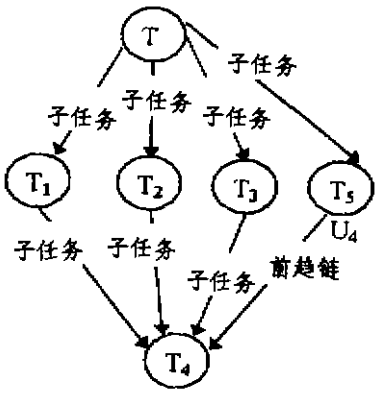

图5 紧耦合工作模式

值得一提的是上述几种模式中 T 起了一个关键的作用，如分离工作模式和松耦合工作模式的区别即是由 T 所控制的，任务中对象版本的唯一性也是由 T 来控制的等。这同样有利于不同工作模式之间的平滑过渡。如在分离工作模式下，若一个用户用包含命令将一个存在于别的子任务中的对象加入到他的子任务，则这些用户 $U_1U_2U_3$ 自动进入到松耦合工作模式，版本合并及版本生成工作完全由任务 T 来控制，它对用户来说是不可见的，当然这时系统可以提供一定的事件通知机制来增强用户之间的了解。在松耦合工作模式下，U_i 可以向 U_jU_k 发出紧耦合工作模式请求，若 U_jU_k 均同意进入紧耦合模式，则系统自动生成一个任务 T_5，将 $T_1T_2T_3$ 中的版本进行合并，以后各用户的编辑均在 T_5 中共享的对象版本之上进行，保证了 WYSIWIS 的原则。

结语 在 KDVer 模型中，一个关键的问题是如何有效地实现同一对象不同版本的合并，当前绝大多数的合并还是手工或半自动化的，这需要深入研究。此外，对于协同工作的有效支持只考虑版本控制是远不够的，因为版本机制虽然能部分解决细粒度的锁机制、事务管理、合作信息的持久保存等，但对于事件通知机制、事件预约等还需作进一步的考虑。

最后，我们必须充分借鉴开放式超媒体系统[9]的功能，利用第三方应用对超媒体文档进行编辑，使得用户使用的工具能与超媒体系统和谐的融合。

主要参考文献

[1] U. K Wiil, &J. J. Leggett, Hyperform: Using Extensibility to develop dynamic, open and distributed hypertext systems, In, The Proc. of the ACM Conf. on Hypertext, ECHT'92 Milano. 1992

[2] D. L. Hicks et al., Version Control in Hypermedia Database, Technical Report TAMU-HRL91-004, Hypertext Research Laboratory, Texas A&M University, July 1991

[3] 李光亚、周学海、赵振西，一种开放式超媒体系统版本模型，软件学报，1998年1月

[4] Nested Composite Nodes and Version Control in Hypermedia Systems. Technical Report, Departmento de Informatica, PUC-Rio de Janeiro, Brasil. August 1993

[5] I. Goldstein and D. Bobrow. A Layered Approach to Software Design. In: Interactive Programming Environments, Mc Graw Hill. 1984

[6] K. Østerbye, Structural and Cognitive Problems in Providing Version Control for Hypertext. Same to [1]

[7] U. K. Wiil and J. J. Leggett, Hyperform: Using Extensibility to develop Dynamic, Open and Distributed Hypertext Systems. In: Proc. of the Fourth ACM Conf. on Hypertext, Seattle, Washington, USA, 1993

[8] M. Melly, Co-operative Working in an Open Hypermedia Environment, Ph. D Thesis. The University of Southampton. Sept. 1995

[9] 李光亚、周学海、龚育昌、赵振西，超媒体系统的开放性探析，计算机科学，24(4)1997