CSCW 系统中的并发控制策略研究

The Analysis of Concurrency Control Mechanism for CSCW System

毛彩辉 于忠党 (渤海大学 信息科学与工程学院 辽宁锦州 121000)

摘要:并发控制是 CSCW 系统的关键技术之一,旨在解决由于多用户同时操作可能产生的冲突。本文首先分析了 CSCW 系统的并发控制问题产生的原因,然后简要介绍了 CSCW 系统中常用的并发控制方法,之后评述了几种典型的并发控制方法的改进策略,并在实例中应用操作转换方法和文档标注方法而对其进行了对比分析。

关键词:并发控制 CSCW 分布式系统 共享对象

1 引言

计算机支持的协同工作 CSCW(Computer Supported Cooperation Work)是一种将人类合作行为模式与计算机支持技术融为一体的新兴技术,即在计算机技术支持的环境下,一个群体协同完成一项共同的任务。CSCW 支持一组用户参与一个共同的任务,并提供给它们访问共享环境的接口,即一个任务,多个用户,多个用户为完成一项共同的任务而组成一个用户群,CSCW 为这个用户群提供协同支持。因此,CSCW 通常也被称为群件(Group ware)。

CSCW 的目标是为一组用户完成一个共同任务而提供协同工作环境。在实时的 CSCW 环境中,必然存在多个用户都需要访问的共享对象(如协作文档、电子白板和协同绘图系统中的图元等),系统必须对用户访问对象的操作做出实时的响应,以使对象的变化与用户的期望相一致。由于可能存在多个用户同时对一个对象的并发访问,冲突就不可避免。而多机之间网络传输时延和不可靠性,以及多用户协同之间人的因素,又给 CSCW 的冲突添加了新的可能性[1]。在对象同时被多个用户并发访问的情况下,服务器如何处理针对共享对象的并发操作以保持各用户端上对共享对象一致是并发控制研究的关键问题。并发控制是 CSCW 系统的关键技术之一,旨在解决由于多用户同时操作可能产生的冲突。

本文首先分析了 CSCW 系统的并发控制问题产生的原因,然后简要介绍了 CSCW 系统中常用的并发控制方法,之后评述了几种典型的并发控制方法的改进策略,并通过应用实例对操作转换方法和文档标注方法进行了对比。

2 并发控制问题产生的原因

在实时协同工作中,正在协同的地理上分散的多个站点可能产生相互独立的操作,同时作用于同一共享应用,它们通过交互事件达到共享应用的一致性。应用程序分布在每个参与者的站点上,站点之间的同步依靠相互传递控制事件(直接或间接)来完成,但是,事件交换存在着潜在的冲突和时延,事件到达和执行可能失序;或由于多个用户实时地对共享的虚拟工作空间进行协同操作时,有两个以上的用户同时操作同一个对象,而操作的意义却不同,致使不同用户的虚拟工作空间相互不一致,引起错误的发生。例如,一个用户删除了一对象,而另一用户却想要移动该对象。

产生并发冲突后,在一致性管理上就会存在不一致性问题:事件之间的因果关系被破坏,最终执行结果不一致。为此,需要提供一定的并发控制策略解决上述问题,以便保证多个用户合作的顺利进行。

3 CSCW 系统中的并发控制方法

在 CSCW 系统中,并发控制是其中的关键技术之一,对并发控制算法的要求包括响应性和实时性。传统的算法(即运用与分布式系统和数据库中的并发控制算法)通过限制并发来保持一致性,影响了群组活动的灵活性,所以这种方法总体上不理想。在实时的

CSCW 系统，并发控制机制在维持共享对象或共享工作空间的一致性的同时，能使用户实时感知到其他协同用户的操作。目前常用的 CSCW 并发控制机制有加锁法、集中控制法、依赖检测、可逆执行、操作转换等。

(1) 加锁法。加锁法是保证数据一致性的常用手段，它提供对共享数据的加锁(lock)和解锁(unlock)，用于控制用户的并发操作。这种方法设计和实现较简单，但它用于 CSCW 系统存在一定缺陷。首先是申请和释放的开销会影响用户的快速反馈，并且系统难于确定何时申请和释放锁。其次，锁的粒度难于确定，过小加重系统负担，过大影响用户操作的并行性。

(2) 集中控制法。采用一个集中控制进程，管理用户对共享对象的操作。所有操作首先发送到集中控制进程，再由它向其他用户分发。它能保证系统操作严格有序，但集中控制进程会成为系统的瓶颈，即当集中控制进程出错时，系统也随之瘫痪，可见集中控制进程削弱了系统的稳定性。另外，操作必须经过集中控制进程的分发，使响应速度变慢，并且由于和服务器通信，会消耗有限的网络带宽。

(3) 依赖检测。依赖检测是用于实时多用户并发控制的一种方法。它对每个操作给出时间戳，根据操作的时间戳检查多个操作之间是否存在冲突。这种方法无需同步机制，本地操作立即执行，没有冲突的操作被接受后也立即执行，因此响应时间很快。但有冲突的操作需要用户的手工干预，用户的错误将导致数据的不一致性。

(4) 可逆执行(UNDO/REDO)。操作可以立即执行，但与操作有关的信息被保存起来，以便在必要的时候取消该操作。在全局时序的作用下，当几个操作冲突时，其中的一个或几个操作被取消并按预定的次序重做一遍，有很好的响应性能，但全局时序难于实现。且一旦出现 REDO 的情况，则系统开销太大。

(5) 操作转换。是依赖检测的变形，即冲突自动消除而不需要手工干预，有较好的响应性能。当操作请求提出时，用户操作立即在本地执行，同时把操作与状态向量一起广播给其他用户。状态向量表明编辑器最近接收到的从其他站点发来的操作序列，每个用户的编辑器有它自己的状态向量。其他用户收到后，把状态向量与本地状态向量比较，决定直接执行(无冲突)或进行转化(有冲突)。

Saul Greenderg 把上面介绍的并发控制方法概括为加锁法和可串行化法两类机制[2]，即除第一种加锁法外，其余均为可串行化法。另外，根据两种方法实现的乐观程度，Saul Greenderg 等还将每种方法分为乐观的和悲观的方法。

4 几种典型并发控制方法及其改进策略

并发控制是 CSCW 系统的关键技术之一，且在技术上具有挑战性。近年来国内不少学者对其进行了深入的研究，使并发控制方法不断得到改进、发展与完善。

Xu Baomin 等人基于有服务器的中心控制法和 UNDO/REDO，提出了一种冲突消解算法[3]。在该算法中，事件的逻辑顺序有 time(全局时间)和 sequence(顺序号)共同决定。这个算法逻辑清晰，但全局时间实现起来比较困难。

在文献[4]中，提出了一种适用于复制结构的 CSCW 并发控制框架模型。算法采用向量时间方法，在规定操作执行次序方面定义一种新的字典全序，并设计了通用失序纠正策略。其基本思想是：为各站点及消息标注向量时间；在操作集上定义一种与向量时间保持一致的全序；各站点根据全序及利用操作的历史记录使最终执行效果与所有操作的全序执行的效果相同。此策略对执行操作集合结果的一致性问题提供了一种有效的解决方法，但对执行过程中次序需求的研究还不完善。

协同编辑系统作为一类典型的多用户交互系统，其并发控制通常采用的方法是操作转换(OT, Operational Transformation)。典型的操作转换有 dOP 转换(dOPT, distributed Operational Transformation)、包含转换(IT, Inclusion Transformation)、删除转换(ET, Exclusion Transformation)、前向转换(FT, Forward Transformation)、后向转换(BT, Backward Transformation)等。

在现有的并发控制方法中，分布式操作转换方法 dOPT 极有希望实现“自然、自由交互”的目标[5]。它通过立即执行用户的本地操作以保证用户界面良好的响应速度；通过“状态向量”来保证操作之间的因果依赖关系；通过操作转换函数以保证对象一致性及满足用户操作意图。文献[6,7]针对 dOPT 算法存在的不足，进行了相应改进。

清华大学杨光信等人基于一种扩展的 ODMG 对象模型给出了一种并发控制模型 oodOPT[6]，其主要思想是利用对象本身及其上所定义的操作语义进行冲突解析。oodOPT 可维护多个对象并为每一个对象维护一个单独的操作日志，其中按执行的先后顺序保存所有能够改变此对象状态的那些原子操作。另外，用户界面上的操作将被转化成被操作的对象上的方法调用。但 oodOPT 对于特定应用的限制条件的一致性维护仍然存在一定的困难。

杨武勇等人[7]在 dOPT 等操作转换算法的基础上提出了基于协作对象数据属性及其操作语义的并发控制框架 madOPT。该方法利用分层存储协作对象的操作日志以减少并发控制服务过程中的查找次数，并结合数据传输框架，利用 Agent 把协作对象及其操作方法的定义加载到并发控制服务中，提高了系统并发控制服务的可扩展性。madOPT 有效地解决了系统运行效率的问题，但是对于 Undo 等操作的并发意愿冲突问题并不能完全公平地解决，仍依靠优先级来解决争端。另外，实时协作系统对本地响应速度的高度敏感，该方法的层次处理模式仍然不完善。

在实时的 CSCW 环境下，操作意愿一致性维护是协同系统一致性维护的重要方面。基于对操作意愿的理解，Sun 等人提出了对象复制概念及算法。刘新福等人[8]根据 CSCW 的特点和用户的操作意愿，提出了一种基于前向变换的串行化并发控制方法，重构对象的操作历史记录，以保持在每一个站点上对象的历史记录的一致性，从而保持共享视图的一致性。此算法对于处理分布式 CSCW 系统中共享对象的并发访问具有实际意义。

值得一提的是文献[9]立足于共享文档本身给出了一种文档标注方法来解决用户期望和操作结果一致性问题，文档标注方法是维护操作意愿的一种新方法。其主要思想是无论某一操作执行前发生了多少并发操作，通过对共享文档加标注的手段，把并发操作执行所引起的文档变化部分屏蔽起来，使该操作执行时的文档状态仍然与该操作产生前瞬间的文档状态一致，从而实现操作意愿的维护。具体方法是：设操作 O 产生前瞬间的文档状态为 DOC，当 O 传送到远程结点执行时，若有 O 的并发操作在该远程结点上已经先执行，则通过对文档加标注的手段，隐藏由于先执行的并发操作所引起的文档的变化部分，使得该远程结点的文档状态仍然是 DOC，然后执行 O，最后，去除所加的标注，以恢复隐藏的文档部分。文档标注方法具有易于理解、通用的特点，算法设计也比较直观，但还很不完善。

王名悠等人提出了一种基于隐藏恢复机制的操作变换算法 FOPT[10]。FOPT（Folding Operation Transformation）算法综合了操作转换和文档标注算法的优点，采用线性文档模型，只对操作本身进行变换。对操作位置进行变换时，不采用转换函数对操作进行两两转换，而是通过维护一个特殊的操作历史记录，使用文档标注算法中隐藏恢复机制来计算并发操作情况下操作的正确位置，从而维护用户的操作意愿。FOPT 算法与原来的算法相比，在时间复杂度和空间复杂度方面做了相应的优化。

余文芳等针对实时图案协同设计中冲突问题，基于原型系统 CoDesign 提出了冲突的预防和检测方法，即基于冲突操作锁定的冲突解决方法。当站点检测到操作发生冲突，保存冲突操作修改的对象的属性值，同时禁用与冲突操作相同类型的操作。站点用户可以继续对冲突对象的其他属性值进行修改，其他站点对该对象有修改权的用户也能协同地对该对象进行编辑。等到所有冲突站点都检测到冲突时，系统禁止冲突对象的操作，由协同设计用户协商选择最终对象的版本来解释冲突。此方法能有效地保护协同用户的设计意图，使得在分布式计算环境下协同设计的冲突问题得到了较完善的解决。

现有的各种群件应用使用了不同的并发控制方法。在实际的系统设计中，设计者应综合考虑各方面因素，根据实际情况需要来设计和选用合适的并发控制策略。

5 应用操作转换方法和文档标注方法的对比分析

本文对操作转换方法和文档标注方法在实时协同文本编辑系统的并发控制机制中的应用作对比分析。实时编辑系统通常采用全复制式的体系结构，即共享对象在所有结点都分别复制一份。对于一个实时协同文本编辑系统存在两个基本编辑操作，即插入和删除。

(1) insert(S,P)，表示在位置 P 之前插入字符串 S。

(2) delete(N,P),表示删除从位置 P 开始的 N 个字符。

并且记第 1 个字符的位置为 0。

如图 1 所示,假设共享文档的初始状态为"ABCDEFGH",O1 和 O2 分别是在站点 0 和站点 1 上同时产生的操作,即 O1 和 O2 是并发操作。设 O1 = insert("aa",4),即表示在"ABCD"与"EFGH"之间插入"aa",O2 = delete(3,3),即表示删除"DEF"。

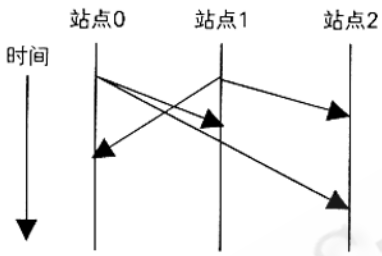

图 1　实时协同编辑系统的操作关系情景

应用操作转换方法:

最基本的操作转换方法 dOPT 中每个站点都维护一个 n 维向量 SV[i],i∈{1,K,n},其中 n 为协作站点个数,每一项对应一个站点。任意站点 k 的状态向量每一个元素 $SV_k[i]$ 表示站点 k 已执行的来自第 i 个站点的操作的个数。为保证复制对象的一致及各操作在各站点的执行结果的一致,每一个操作在被任一站点实际执行前,dOPT 用一系列"转换函数"将它同历史记录中的那些操作进行转换,即调整操作之间的位置参数。当转换后的操作被执行之后,它将被加入到历史记录中,同时状态向量的相应分量将被加 1。

O_1 在站点 0 执行后,文档变为"ABCDaaEFGH",O_2 到达站点 0 时,通过操作转换函数转换后,得到两个操作 delete(1,3) 和 delete(2,6),删除"DEF",得到正确的执行结果"ABCaaGH"。

应用文档标注方法:

文档被标注部分的类型有两种[9]:即不可见和可见但实际不存在。不可见表示从操作的角度看,这部分文档是不存在的,但实际上是存在的;可见但实际不存在表示从操作的角度看,这部分文档是可见的,但实际上已经被前面执行的某个操作删除了。本例用下划线代表"不可见"标注,用删除线代表"可见但实际不存在"标注。

在站点 0,O_1 执行后,文档状态变为"ABCDaaEFGH"。O_2 到达站点 0 后,需要屏蔽 O_1 所引起的文档变化部分,即对"aa"加下划线标注,文档状态为"ABCDaaEFGH"。执行 O_2,删除位置 3 开始的 3 个字符,由于"aa"带有下划线,不在删除之列,因此删除"DEF",得到"ABCaaGH",然后去掉"aa"的下划线标注,得到正确结果"ABCaaGH"。

在站点 1,O_2 执行后,文档状态显示为"ABCGH",但并不实际删除"DEF",而是在"DEF"各字符上标明操作 O_2,所以文档状态仍为"ABCDEFGH"。O_1 到达本站点后,需屏蔽 O_2 所引起的文档变化部分,即对"DEF"加删除线标注,文档状态为"ABCDEFGH"。执行 O_1,在位置 4 之前插入"aa",得到"ABCDaaEFGH",然后去掉"D"、"EF"上删除线标注,得到正确结果"ABCaaGH"。

通过以上两种方法的应用,可以看出,操作转换方法着眼于操作本身,试图通过变换操作的参数来实现操作意愿维护,具有较好的响应效果,但由于其算法与操作语义紧密相关,以及网络传输延迟的不确定性,当结点规模稍大点时,操作之间的关系变得非常复杂,操作转换算法设计的难度和技巧性要求很高。上面的应用中,采用操作转换方法时,操作 O_2 变换后得到两个操作,而采用文档标注方法,操作 O2 不需作任何变化,执行时只需记住带下划线标注的字符不在删除之列。可见文档标注方法易于理解,算法设计也较直观,但并不完善。

6　结束语

本文对 CSCW 系统的并发控制策略进行了综述,首先分析了 CSCW 系统的并发控制问题产生的原因,简要介绍了 CSCW 系统中常用的并发控制方法,之后评述了几种典型并发控制的方法的改进策略,并通过应用实例对操作转换方法和文档标注方法作了对比分析。CSCW 系统中的并发控制机制与传统的分布式系统有很大的区别,尽管传统的并发控制可以满足多用户及分布性的要求,却满足不了 CSCW 系统所特有的特性,如群件之间的感知性。因此,传统的并发控制方法不完全适合于 CSCW 系统的要求,如何对传统的并发控制方法进行改进,使它满足 CSCW 系统的要求是 CSCW 研究的一个重要方面。　(下转第 93 页)

(上接第 125 页)

参考文献

1 刘新福、王光彩、代雯君等,集中式 CSCW 环境中对实时共享对象的并发控制算法[J],小型微型计算机系统,2002.23(6).

2 Greeberg S and Marwood D. Real Time Groupware as a Distributed System:Concurrency Control and its Effect on the Interface. Proceeding of CSCW'94,1994.

3 Baomin Xu,Hang Shi,Shouxun Lin. Some issues implantation of a whiteboard tool (OB/DL). CSCW98, Tokyo,1998. http://www.researchindex.com.

4 肖波、张东、诸鸿文,计算机支持的协同工作并发控制策略[J],上海交通大学学报,1999.339(1).

5 Ellis C A,Gibbs S J. Concurrency control in groupware systems. Proceedings of ACM SIGMOD Conference on Management of Data,Seattle,1989.

6 杨光信、史美林,全复制结构下基于对象数据模型的并发控制[J],计算机学报,2000.23(2).

7 杨武勇、史美林、姜进磊,一种集成组播代理和操作转换的并发控制方法[J],软件学报,2004.15(4).

8 刘新福、吕钊、代雯君等,分布式 CSCW 环境中并发控制的串行化方法[J],2001.27(10).

9 何鸿君、泉源、罗莉,协同编辑中维护操作意愿的文档标注方法[J],软件学报,1999.10(2).

10 王名悠、王晓斌,实时协同文本编辑系统中共享文档的一致性维护[J],福建电脑,2006(3).