第36卷　第22期
Vol.36　　*No.22*

计　算　机　工　程
Computer Engineering

2010年11月
November 2010

· 特邀论文 ·　　文章编号：1000—3428(2010)22—0001—06　　文献标识码：A　　中图分类号：TP391

实时协同中的一致性维护关键技术

邵　斌，卢　暾，顾　宁
（复旦大学计算机科学技术学院，上海 201203）

摘　要：实时协同广泛采用数据复制技术来隐藏网络延迟、提高响应速度和改善交互体验。在复制式结构中，用户可以无约束地在不同数据副本上进行操作，这必然会引发一致性维护问题。针对上述问题，根据实时协同应用的特点和需求，总结一致性维护研究面临的技术挑战，通过研究实时协同中操作转换、地址空间转换和 WOOT 一致性维护的关键技术，讨论 Undo 和 String 转换等难点问题，分析这些关键技术在单用户软件向多用户协同软件透明转换、Web2.0 环境和移动协同环境中的应用情况。

关键词：实时协同；一致性维护；一致性模型；操作转换；地址空间转换

基金项目：国家自然科学基金资助项目(60736020, 60803118)

作者简介：邵　斌(1982－)，男，2010年复旦大学博士毕业，现任微软亚洲研究院副研究员，主研方向：CSCW，Web信息抽取；卢　暾，博士、复旦大学计算机科学技术学院讲师，主研方向：协同计算；顾　宁，复旦大学计算机科学技术学院教授、博士生导师，主研方向：CSCW中的一致性维护，社会网络的数据分析，云计算平台技术

收稿日期：2010-10-08　　**E-mail：**binshao@fudan.edu.cn, lutun@fudan.edu.cn, ninggu@fudan.edu.cn

Key Techniques of Consistency Maintenance in Real-time Collaboration

SHAO Bin, LU Tun, GU Ning
(School of Computer Science, Fudan University, Shanghai 201203, China)

【Abstract】Data replication is widely used to hide network latency, enhance response time and improve interaction experience in real-time collaboration. Users can operate on data replicas unconstrainedly, which results in consistency maintenance problems of these replicas. Aiming at the problem, this paper introduces the challenges of consistency maintenance according to the features and requirements of real-time collaboration applications. By analyzing representative key techniques of operation transformation, address space transformation and WOOT, it elaborates the challenging problems such as Undo and String transformation, and details how these key techniques are applied in the transparent adaptation from single-user applications to collaborative ones, Web 2.0 and mobile collaboration environments.

【Key words】real-time collaboration; consistency maintenance; consistency model; Operational Transformation(OT); address space transformation

1　实时协同应用的特点和挑战

正在进行的信息革命对传统的生产、生活方式产生了深远的影响。计算机技术的发展使人与人之间的协作可以跨越空间上的隔阂，跨区域的实时交流和协作变得越来越流行。技术的进步和发展使许多原来只能在同一地点进行的高交互性协同活动可以通过由网络连接的协同应用系统在线进行[1-3]。本文讨论的重点是有着高交互性需求的实时分布式协同应用。对于此类协同应用，用户操作的本地响应性是非常重要的一个性能评价指标。

互联网上的网络通信延迟是不确定的，而且通常比较高，尤其是通过无线网络进行联网时。那么，如何在比较高的网络延迟下保证用户操作的高响应性呢？网络通信延迟是客观存在的，除了改善网络基础设施外，对于现有的网络环境而言，网络延迟不可能从系统层面被去除，甚至不可能得到大幅度的削减。既然不能消除网络延迟，就需要用某种机制来隐藏它。在分布式系统领域，一个经典而有效的隐藏网络延迟的方法就是数据复制技术[4]。换句话说，每个客户端拥有一个本地的数据副本，用户的操作可以在本地数据副本上直接执行，使操作的本地响应与网络延迟无关。这样一来，用户就获得了很好的本地响应性。然而，一方面，数据复制技术带来了数据的高可用性以及本地操作的高响应性；另一方面，它又带来了不同用户操作之间的并发控制问题以及不同数据副本之间的数据一致性维护问题。

随着网络技术的发展、Web应用的普及以及多种协同终端以不同接入模式的参与，涌现了一大批以Web2.0和移动计算为代表的新型协同应用。这些应用呈现出大规模、动态性、操作复杂性、数据类型多样性等新特性，给实时协同中一致性维护的研究带来了新的技术挑战。实时协同中的一致性维护技术研究有着20多年的历史，一直是主流CSCW与群组权威国际会议(如ACM CSCW、ACM GROUP)中的研究重点和难点。它的研究对推动实时协同应用的发展和普及起到了重要作用。

本文首先对实时协同中的一致性维护关键技术做一个简单的回顾和总结，然后针对其中的Undo问题、String操作等难点问题进行讨论，最后给出实时协同中一致性维护技术在实际环境和系统中的应用情况。

2　实时协同中一致性维护技术

实时协同应用系统从技术层面上讲，涉及了分布式系统、协同系统、人机交互等多个层面的知识。有别于其他的分布式系统，它有 2 个显著的特点：

(1)分布性，不同的参与者往往地理上是分布的。

(2)交互性和协作性，系统中有人的直接参与，并且参与者与协同系统之间以及参与者和参与者之间有较强的交互性。

针对具有分布性、交互性和协作性的实时协同应用系统，研究人员设计了很多一致性维护的方法。其中，操作转换(Operational Transformation, OT)[5-6]、地址空间转换[7]、WOOT(WithOut Operational Transformation)[8]等乐观的一致性维护技术是这一类一致性维护方法的代表。而基于操作转换的一致性维护算法是目前最为成熟、被广泛承认和接受并得到普遍应用的一种一致性维护方法。

2.1　操作转换技术

2.1.1　操作转换技术概述

操作转换技术是针对上述分布式协同应用需求应运而生的一种数据复制式架构下的乐观并发控制和数据一致性维护方法。操作转换技术允许用户无约束地执行本地操作，而不需要额外地申请锁或操作令牌。因此，即使用户在高延迟的广域网上进行协作，也可以获得很好的本地响应性。同时，操作转换技术会保留所有用户操作的效果，将所有用户操作的效果在不改变其“操作意愿”的前提下进行自动合并，避免了操作之间的覆盖和重写。

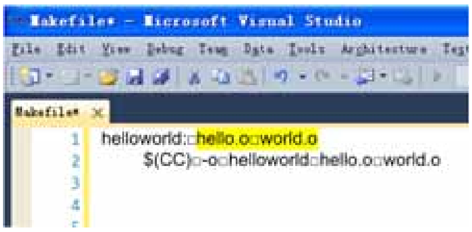
操作转换技术虽不能解决操作之间的“语义”冲突，但是由于它根据“操作意愿”保留了所有用户的操作效果，因此可以简化后续的冲突检测和消解过程。另外，在操作转换技术支持的系统中，客户端站点之间通常采用细粒度的操作更新。相对于基于文档状态的更新同步机制，基于操作的数据更新在大多数情况下可以节省同步时的数据传输量，并减少数据同步时的操作冲突[4]。

操作转换技术特别适用于有高操作响应需求的分布式协同应用中的数据一致性维护问题。在此类分布式协同应用中，用户通常会并发地修改同一个数据对象，这时阻塞的、粗粒度的并发控制和一致性维护方法会严重影响用户的工作效率，通常是不能被接受的[9]。操作转换技术以其无锁、细粒度的乐观并发控制特性成为特别适合此类协同应用系统的一致性维护方法。符合此类应用特点的最简单的应用实例就是组编辑系统(Group Editing System)。而操作转换技术最早也起源于对协同组编辑中并发控制和数据一致性维护问题的研究[5-6]。作为该类分布式协同应用系统的抽象模型，在过去的 20 余年里，组编辑系统一直担任了推动操作转换技术发展和前进发动机的角色[5-6, 8, 10-11]。

2.1.2　操作转换的基本思想

由于不同协同站点之间的操作具有并发性，因此当一个远程站点的操作到达本地站点后，其操作的上下文可能已经与产生该操作的上下文不同。为了正确地执行该操作，操作转换技术在执行该远程操作之前先对其进行转换，使得在当前的上下文下执行该操作而不改变该操作的原始操作效果。

为了更直观地阐述操作转换技术的基本思想，本文考虑一个在协同软件开发中的情景。假设有一个 C 语言项目的 Makefile 片段，如图 1 所示(为了表达清晰，用符号“□”表示空格)。



```
helloworld:□hello.o□world.o
	$(CC)□-o□helloworld□hello.o□world.o
```

图 1　Makefile 片段

在该项目开发的过程中，有 2 个用户 Alice 和 Tom 并发地执行了 2 个操作修改了图 1 中高亮的字符串：hello.o□world.o。Alice 通过操作 O1=ins(0,”foo.o□”)将该字符串更新为：foo.o□hello.o□world.o。同时，Tom 通过操作 O2=ins(8,”bar.o□”)将该字符串更新为：hello.o□bar.o□world.o。如图 2 所示。

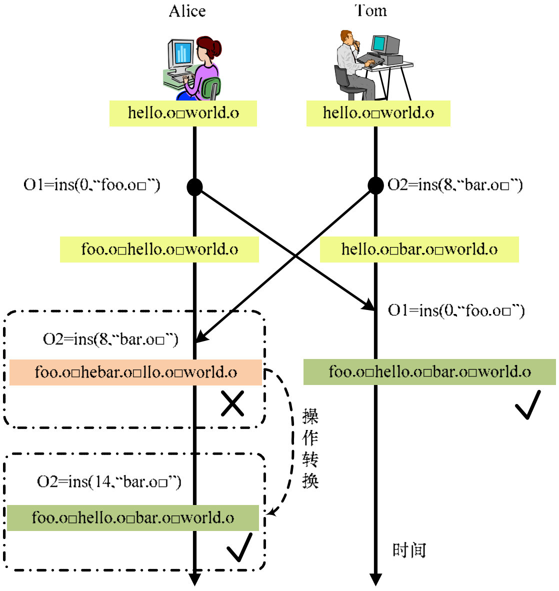


图 2　操作转换的基本思想

用户 Alice 的操作执行情况如下：当操作 O2 到达 Alice 的站点时，如果该操作直接在当前副本执行，Alice 的 Makefile 中，该片段将被修改为：foo.o□hebar.o□llo.o□world.o，由于 Alice 之前已经执行了操作 O1，当 Alice 执行操作 O2 时，O2 操作的上下文已经不是它产生时的上下文，因此直接在当前的上下文下执行操作 O2 会产生错误的结果。在这个场景中，Alice 需要将 O2 转换成 O2’，使 O2’可以在当前的文档状态“foo.o□hello.o□world.o”上正确执行。因为 O1 在 O2 的插入位置之前插入了一个长度为 6 的字符串“foo.o□”，所以需要将 O2 的插入位置右移 6 个操作位置来包含 O1 已经执行操作 O1 的操作效果，得到 O2’=ins(14, “bar.o□”)。这时，在 Alice 的当前状态“foo.o□hello.o□world.o”上执行 O2’得到：foo.o□hello.o□bar.o□world.o。

对于用户 Tom 来说，当收到操作 O1 后，O1 可以直接按其原始形式即 ins(0,”foo.o□”)正确地执行，因为此时 O1 的操作位置不受操作 O2 的影响。在 Tom 当前状态“hello.o□bar.o□world.o”上执行操作 O1 后也可以得到：foo.o□hello.o□bar.o□world.o。这样，Alice 和 Tom 协同编写的程序片段的一致性得到了维护。

2.1.3　操作转换技术的演化

对操作转换技术的研究可以追溯到 1989 年 C.A. Ellis

和 S.J. Gibbs 的开创性工作[5]。自此，学术界对操作转换技术展开了广泛而深入的研究。在至今 20 余年的研究中，一大批操作转换算法被设计出来，比较有代表性的有 dOPT[5]、Jupiter[12]、adOPT[13]、SOCT2[14]、GOT[15]、GOTO[6]、SOCT3/4[10]、SDT[11]、ABT[16]、COT[17-18]等。

早期的操作转换技术研究侧重于设计新的操作转换函数和对操作转换属性的探讨，缺乏操作转换一致性模型(即应当满足什么一致性标准，实时协同系统才是正确的)的探讨。后来，随着研究的深入，出现了一些具有代表性和影响力的一致性模型，包括：

(1)M. Ressel 等提出的 CC 一致性模型

CC 模型[13]包含 2 个方面的一致性标准：因果一致性(Casualty Preservation)和结果一致性(Convergence)。“因果一致性”(因果保持)是指任意操作的执行都不能违反操作之间的产生次序，更准确地讲，是要满足操作之间时序上的先于关系(Happened Before)[19]。即如果一个操作 O1 先于另一个操作 O2 产生，那么在任何站点上都要先于 O2 执行 O1。“结果一致性”则是系统在执行所有操作之后，不同数据副本结果是一致的。实际上，CC 一致性模型是分布式系统所需要遵循的正确性约束标准。

自 1996 年提出以来，大多数的 OT 算法都是基于该模型而开发的。CC 模型的定义仅仅考虑了结果的收敛性，但对于收敛到什么样的一致性结果才是合理和正确的并未做约束。

(2)Sun Chengzheng 等提出的 CCI 一致性模型

如上文所述，随着对操作转换技术理解的加深，人们逐渐发现 CC 模型不足以完整约束一个操作转换系统的行为。Sun Chengzheng 等在 1998 年提出了一个在很长的一段时间内被人们广泛接受的一致性模型——CCI 一致性模型[15]，它对操作转换一致性的研究与发展有着深远的影响。CCI 一致性模型的主要贡献是在 CC 模型的“因果一致性”和“结果一致性”的基础上增加了一个正确性约束条件：意愿一致性(Intention Preservation)。该条件的核心思想是：任何操作的执行都不能违反操作产生时的原始操作意愿。“意愿一致性”是在实时协同中首次尝试明确地限定收敛性。

(3)Li Du 等提出的 CA 一致性模型

Li Du 等提出了一个严格形式化论述和证明的一致性模型 CA 模型[16, 20-21]。CA 模型从理论上形式化了 CCI 模型所做的正确性约束。CA 模型包括 2 个一致性标准：因果关系保持(Causality Preservation)和 Admissibility 属性保持。其中的 Admissibility 属性保持间接蕴含了一致性约束条件，因此，在 CA 模型中没有直接出现“结果一致性”(Convergence)的约束。在 CA 模型中，除了因果关系保持和一致性约束之外，还通过 Admissibility 属性保持严格限定了操作转换的行为。CA 模型主要的贡献是使基于该模型所设计的操作转换算法可以进行形式化的描述和论证。

2.2　地址空间转换技术

文献[7]提出了一种基于地址空间转换(Address Space Transformation, AST)的一致性维护技术。该技术与操作转换技术具有相同的技术目标，即支持实时的协同工作，并自动维护不同协同站点之间的数据一致性。为了正确地执行一个远程操作，操作转换技术在每次执行操作之前需要先对操作进行转换。与操作转换技术相比，地址空间转换的方法不对操作本身进行转换，而是对操作执行的地址空间进行转换。

仍然考虑 2.1.2 节的例子。

当用户 Alice 执行了操作 O1 之后，程序片段的状态是“ foo.o□hello.o□world.o ”。如果在这个状态上直接执行操作 O2，将会导致错误的程序片段“ foo.o□hebar.o□llo.o□world.o ”。操作变换方法是通过把操作 O2=ins(8,”bar.o□”)变换成 O2’=ins(14,(bar.o□))，找到正确的插入位置，从而得到正确的程序片段。与操作变换不同，地址空间变换不对操作 O2 本身进行转换，而是对操作执行的地址空间进行转换，即把“ foo.o ”标记为无效(“ foo.o ”)，然后把状态回溯到 O2 产生时的程序片段状态“ foo.o□hello.o□world.o ”来执行操作 O2，从而得到“ foo.o□hello.o□bar.o□world.o ”，最后把“ foo.o ”还原为有效，得到一致的结果“ foo.o□hello.o□bar.o□world.o ”。如图 3 所示。

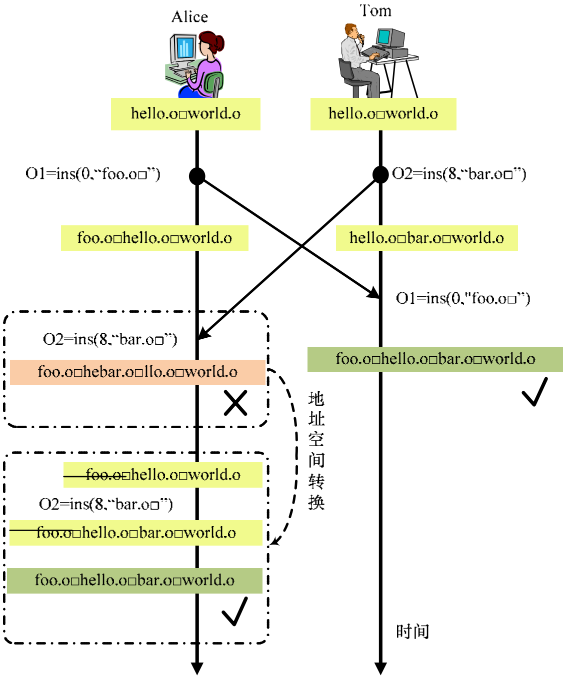


图 3　地址空间转换的基本思想

从以上分析可以看出，地址空间转换方法的原理是：在每个操作执行之前，通过对其地址空间进行转换，将当前的执行上下文回溯到该操作产生时的状态，使得该操作相当于在其原始的操作上下文下执行。通过这种方式，地址空间转换技术可以保证每次操作的执行都能保持其原始的操作效果。这个过程涉及 2 个关键技术 标记(Mark)和回溯(Retrace)，前者用来标记一个字符节点在某个给定时间是否有效，后者则用来把文档的地址空间回溯到某个操作产生时的地址空间。所以，地址空间转换技术又称为 Mark & Retrace 技术。

地址空间转换技术能克服传统操作转换技术在满足 CCI 模型方面的困难性和复杂性，并克服了传统方法在解决 dOPT[5]和 false-tie(ERV)[11,15]等难题时遇到的问题。

2.3　WOOT 方法

文献[8]提出了一种称为 WOOT 的一致性维护方法，该方法与地址空间转换的方法有异曲同工之处。在 WOOT 方法中，算法为每一个数据对象分配一个全局唯一的 ID，其原子操作的定义建立在这些全局 ID 基础之上。因为每个操作的数据对象都有全局唯一的 ID，如果一个远程操作的数据对象已经存在，就可以根据其全局 ID 直接找到该数据对象并执行其对应的操作；如果一个远程的数据对象还未存在，则可以通过一个递归的过程逐步求精地找到其正确的插入边界。同时，

由于使用全局 ID，因此 WOOT 方法不是采用基于操作位置的寻址方式[22]。

与绝大多数操作转换算法和地址空间转换方法不同的是，WOOT 放松了对操作之间时序上的先于关系的约束。在操作转换算法和地址空间转换方法中，任何一个操作在执行前，时序上先于该操作的前继操作必须已经在该站点上被执行。而 WOOT 放松了该要求，只要该操作直接依赖的操作已经被执行(在 WOOT 中称为满足前置条件)，该操作就可以在当前上下文执行。

3　实时协同一致性维护技术中的 Undo 问题

Undo 是交互式应用程序中的一个重要功能，许多日常的单用户应用程序，如文本编辑工具、编程工具、设计软件、绘图软件、Web 浏览器，都允许用户按时间顺序撤消最近的一个或多个历史操作。在这些应用中，Undo 通常用于用户层面的错误恢复，例如修改拼写错误。同时，由于 Undo 机制能撤消错误操作所带来的不良影响，因此对鼓励用户探索不熟悉的应用程序功能起着积极的作用[23]。

与单用户应用程序中的 Undo 机制相比，在协同技术与应用中实现 Undo 机制是一个极具技术挑战的工作[18, 24-25]。当分布在不同地方的用户进行协同交互时，本地和远程操作可能会由于并发性被任意地交叉执行。而用户发出的 Undo 操作既可能是指撤消最后一个本地操作也可能是指撤销最后一个在本地执行的操作(可能是本地产生操作，也可能是远程站点产生操作)。如果用户要撤销的是最后一个本地操作，可能该操作执行后又有其他远程操作已经在本地执行过。因此，该操作所定义上下文已不是当前的文档状态，撤销该操作也就不能靠简单地执行其逆操作来完成。当用户要撤销最后一个远程操作时，系统需要正确判定要撤销的是哪个用户执行的哪个操作；否则，该 Undo 操作在不同的站点上可能会有不同的执行效果，引起不可预测的系统行为[23]。特别地，在分布式协同中要能达到撤消历史操作队列中任何所选操作的操作效果，即支持“选择性 Undo”(Selective Undo)或“Undo 乱序执行”(Undo Any Operation at Any Time)的功能。

由于 OT 技术可以对任意一对操作进行转换以交换其执行顺序，因此广泛应用于协同系统中来实现 Undo 功能。但相关研究表明，构造一个完全满足 CCI 模型的操作转换控制算法是非常困难和复杂的，而且大部分基于 OT 的 Undo 解决方案都难以对上述“选择性 Undo”或“Undo 乱序执行”提供完全的支持[13, 26-27]。其主要原因是，OT 方法无法获得每个操作产生时的用户状态，从而导致难以操作意愿。而如上文所述，AST 方法通过恢复到操作产生时的地址空间状态来消除其他并发操作的影响，找到正确的执行位置，从而正确地维护操作的意愿。在基于 AST 的 Undo 问题解决方案中，首先将 Undo 操作映射到每个对象节点，对于一个节点上多个冲突的操作意愿，AST 方法先将它们进行分离，然后分别考察这些操作产生时的用户状态，以理解它们的操作目的，从而实现对 Undo 操作的正确判断和处理。分析表明，基于 AST 的 Undo 解决方案能够很好地解决 Ambiguous Reference、Insert-insert-tie Dilemma、Do-Undo-pair Trap、Overlapping Deletes 等 Undo 难题，与传统基于 OT 的方法相比，其复杂度也较低[27]。

现有基于 OT 的 Undo 解决方案没有完整地证明算法所依赖的正确性条件，同时完整地支持对任意操作的选择性 Undo 开销比较大。针对上述问题，文献[28]提出了一种新的基于 OT 的 Undo 解决框架 ERU，它能够将 Do 和 Undo 操作统一到一个框架下进行高效处理。其基本思路是将站点历史操作队列中的所有操作按照操作效果序排列，不论本地产生一个普通的 Do 操作或者是对某个历史操作的 Undo 操作，本地线程都将对它进行统一的处理和广播。分析表明，ERU 满足 CA 一致性模型，也能够解决上述经典 Undo 难题，同时支持选择性 Undo，并能将 Do 和 Undo 的处理复杂度优化为线性。

4　面向 String 的实时协同一致性维护技术

上文讨论的一致性维护方法主要还是考虑线性数据结构方面面向单个操作对象上的原子操作(如对字符的添加和删除)，而在真实的系统中，基于对象串(String)的操作是非常常见的，比如文本编辑中的剪切和粘贴操作(这里的 String 是指对原子对象“串”的抽象，而不是特指一般意义上的字符串)。由于 String 操作之间有非常复杂的重叠关系，而且操作在转换过程中可能发生级联分裂，因此 String 转换算法非常复杂。针对上述问题，文献[29]提出了一种支持 String 操作的一致性维护算法 ABTS，它通过一种“双递归”的操作转换机制，简洁、高效地处理操作转换中的级联分裂。同时 ABTS 满足 CA 一致性模型，并得到了严格的形式化证明，它能有效避免算法应用中的各种难题。

进一步的研究发现，由于大部分面向单个操作对象的操作转换算法都需要平方级或者更长的时间来整合单个远程的字符串操作。随着历史队列长度的增加，复杂度高的算法很容易使应用的相应时间超过 100 ms 的本地响应阈值。针对这些问题，文献[30]基于 ABTS 设计了一个精简、优化的 String 转换算法 ABTSO。它在保留 String 转换主体思想的同时，简化了 String 转换算法进行删除操作时的复杂细节，即在删除操作进行任何转换之前先把面向 String 的删除操作拆分为字符操作。利用操作间的操作效果关系，ABTSO 算法的时间复杂度优化为线性。

5　实时协同一致性维护技术的应用

5.1　单用户软件透明转换成多用户协同软件中的应用

文献[31-32]提出了一种称为透明适配(Transparent Adaptation, TA)的方法，该方法能够把已有的单用户软件或应用变成多用户的实时协同软件，而不需要修改原有程序的源代码。基于 OT、AST 等一致性维护技术以及单用户应用编程接口(Application Programming Interface, API)与不同一致性维护技术的数据和操作间的适配方法，是支撑 TA 方法的两大基石。同时还提出了一种支持 TA 方法的通用的协同系统体系结构，如图 4 所示[24]。

图 4　通用协同系统体系结构

该通用体系结构包括 3 个组件：单用户应用程序(Single-user Application, SA)，协同适配器(Collaboration Adaptor, CA)和通用协同引擎(Generic Collaborative Engine, GCE)。SA 可以是任何单用户应用，提供传统单用户接口的特性和功能。下层的多用户协同对 SA 来说是透明的。CA 作为 SA 和 GCE 之间的连接纽带，负责调用 SA 提供的 API 来捕获并重现用户操作指令，并提供应用相关的协同能力。该组件既能感知单用户应用，又能感知多用户协同。GCE 提供应用无关的协同能力，它封装了一组实时协同相关技术(如 OT、AST 等支持实时协同的一致性维护技术、群组 Undo 技术)。该组件能够感知多用户协同，但独立于单用户应用。

从实现的角度来说，在 SA 和 GCE 中使用 CA 的目的是为了隐藏与实际应用相关的一些问题，从而能够实现对 GCE 的独立调试与测试，提高 GCE 的可重用性，减少透明适配的复杂度。

目前已有不少基于该体系结构而实现的实时协同软件，如支持日常协同办公的 CoWord[31-32]、CoPowerPoint[33]、支持协同工程设计的 CoAutoCAD[34]、支持三维动画协同设计的 CoMaya[35]、支持协同程序设计的 CoEclipse[36]。

5.2 Web2.0 环境中的应用

随着网络与 Web 技术的发展，出现了一种以用户为中心、强调用户贡献与协同交互的新型实时协同环境——Web2.0 环境，如图 5 所示。由于该环境中参与实时协同的用户数量巨大，因此很多 Web2.0 站点通常采用镜像站点的方式支持大规模的访问量和实现负载均衡，并且协同交互的数据大多是用 XML 结构来描述的。除了支持一般用户的非事务模型，Web2.0 站点管理员或资深用户可能需要用事务模型执行一些重要的操作以对站点进行维护和管理。而传统方法无法满足 Web2.0 协同应用中一致性维护技术对 XML 结构数据和混合操作类型等方面的新需求。

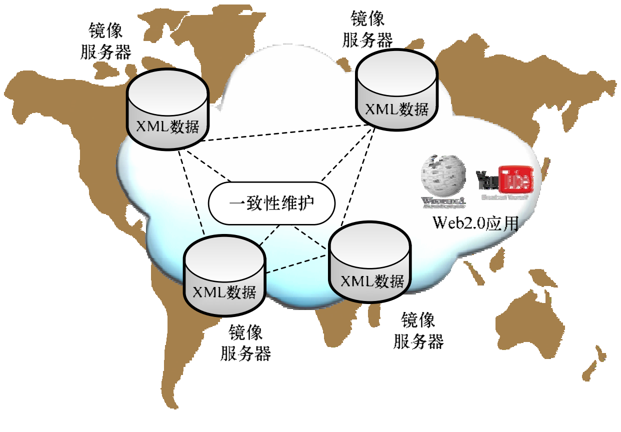


图 5 Web2.0 实时协同环境

通过对 AST 方法进行扩展，文献[37]提出了一种 Web2.0 应用中的无锁一致性维护方法。它将 XML 数据表示成倒排表的结构，在查询过程中利用节点上的历史操作信息判断节点的状态；对于可能出现的事务操作，通过维护每个站点产生该事务操作对应的 NOOP 操作与其他普通操作的先序依赖关系，将对普通操作和事务操作的处理统一到一个框架下，利用地址空间转换思想进行一致性维护。实验结果表明，相比传统的两阶段锁、序列化等方法，该方法不依赖于站点间的网络延迟，并且能更好地实现负载平衡，满足操作的实时响应。

5.3 移动协同环境中的应用

手机、PDA 等移动智能终端在人们的日常生活中扮演着越来越重要的角色，并成为人们协同交互的重要工具。但是，移动设备的计算能力、电池续航能力都比较有限，其无线连接也呈现出不稳定的特点，很多时候移动设备由于弱连接而累积的大量离线操作序列需要统一进行处理，如图 6 所示。现有单个操作进行变换的一致性维护方法扩展到基于操作序列变换时产生的较高的处理复杂度导致其不适用于计算能力和续航能力有限的移动协同环境。

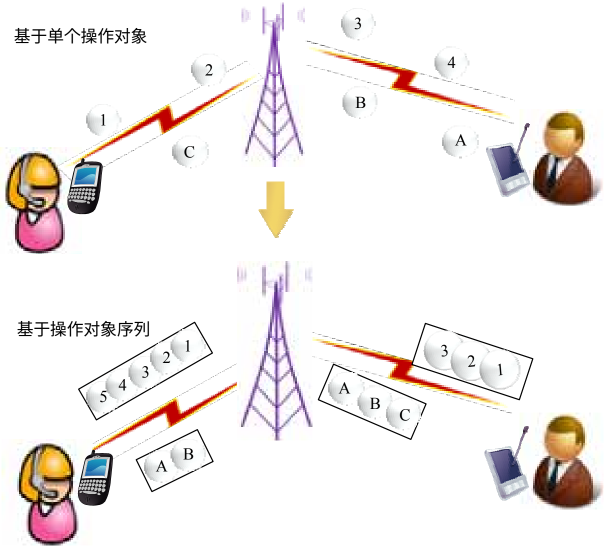


图 6 移动实时协同环境

针对移动环境的新需求和新特点，文献[38]提出了一种高效的基于序列变换的一致性维护算法 ABST。其主要思想是引入操作间的操作效果序，并利用有序的操作效果序进行子操作合并以减少同步操作的转换频率。该算法将目前主流的立方级复杂度的序列操作变换算法改进成线性，从而大幅提高了一致性维护的效率，非常适用于移动实时协同环境。

6 结束语

实时协同技术最大的技术挑战是如何在隐藏网络延迟满足分布的用户对实时协同高响应需求的同时，又能方便、有效地维护数据对象的一致性。随着网络与协同技术的进一步发展，实时协同技术将在云计算、物联网、社会网络等各种新的环境中有着广阔的应用前景。针对这些新的协同环境将给实时协同的一致性维护带来新的挑战，需要有新的模型、算法来维护各种数据对象的一致性。实时协同中的一致性维护技术是一个不断发展并有着活力的研究领域，还需要学术界和工业界更加深入地研究与探讨。

参考文献

[1] Joslin C, Molet T, Magnenat-Thalmann N. Advanced Real-time Collaboration over the Internet[C]//Proc. of the ACM Symposium on Virtual Reality Software and Technology. New York, USA: ACM Press, 2000.

[2] Olson G M, Zimmerman A, Bos N. Scientific Collaboration on the Internet[M]. [S. l.]: The MIT Press, 2008.

[3] Jara C A, Candelas F A, Torres F, et al. Real-time Collaboration of Virtual Laboratories Through the Internet[J]. Computer Education, 2009, 52(1): 126-140.

[4] Saito Y, Shapiro M. Optimistic Replication[J]. ACM Computing Survey, 2005, 37(1): 42-81.

[5] Ellis C A, Gibbs S J. Concurrency Control in Groupware Systems[C]//Proc. of SIGMOD'89. New York, USA: ACM Press, 1989.

[6] Sun Chengzheng, Ellis C. Operational Transformation in Real-time Group Editors: Issues, Algorithms, and Achievements[C]// Proc. of CSCW'98. New York, USA: ACM Press, 1998.

[7] Gu Ning, Yang Jiangming, Zhang Qiwei. Consistency Maintenance Based on the Mark & Retrace Technique in Groupware Systems[C]//Proc. of GROUP'05. New York, USA: ACM Press, 2005.

[8] Oster G, Urso P, Molli P, et al. Data Consistency for P2P Collaborative Editing[C]//Proc. of CSCW'06. New York, USA: ACM Press, 2006.

[9] Hymes C M, Olson G M. Unblocking Brainstorming Through the Use of Simple Group Editor[C]//Proc. of CSCW'92. [S. l.]: ACM Press, 1992.

[10] Vidot N, Cart M, Ferrie J, et al. Copies Convergence in a Distributed Real-time Collaborative Environment[C]//Proc. of CSCW'00. New York, USA: ACM Press, 2000.

[11] Li Du, Li Rui. Preserving Operation Effects Relation in Group Editors[C]//Proc. of CSCW'04. New York, USA: ACM Press, 2004.

[12] Nichols D A, Curtis P, Dixon M, et al. High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System[C]// Proc. of UIST'95. Pittsburgh, Pennsylvania, USA: ACM Press, 1995.

[13] Ressel M, Nitsche-Ruhland D, Gunzenhauser R. An Integrating, Transformation-oriented Approach to Concurrency Control and Undo in Group Editors[C]//Proc. of CSCW'96. New York, USA: ACM Press, 1996.

[14] Suleiman M, Cart M, Ferrie J. Concurrent Operations in a Distributed and Mobile Collaborative Environment[C]//Proc. of ICDE'98. Washington D.C., USA: IEEE Computer Society, 1998.

[15] Sun Chengzheng, Jia Xiaohua, Zhang Yanchun, et al. Achieving Convergence, Causality-preservation, and Intention Preservation in Real-time Cooperative Editing Systems[J]. ACM Transactions on Computer-Human Interaction, 1998, 5(1): 63-108.

[16] Li Rui, Li Du. Commutativity-based Concurrency Control in Groupware[C]//Proc. of CollaborateCom'05. San Jose, USA: [s. n.], 2005.

[17] Sun David, Sun Chengzheng. Operation Context and Context-based Operational Transformation[C]//Proc. of CSCW'06. New York, USA: ACM Press, 2006.

[18] Sun David, Sun Chengzheng. Context-based Operational Transformation in Distributed Collaborative Editing Systems[J]. IEEE Transactions on Parallel Distributed System, 2009, 20(10): 1454-1470.

[19] Lamport L. Time, Clocks, and the Ordering of Events in a Distributed System[J]. Communications of ACM, 1978, 21(7): 558-565.

[20] Li Rui. Operational Transformation-based Concurrency Control in Group Editors[D]. Texas, USA: Texas A&M University, 2006.

[21] Li Du, Li Rui. An Admissibility-based Operational Transformation Framework for Collaborative Editing Systems[J]. Computer Supported Coopertive Work, 2010, 19(1): 1-43.

[22] Oster G, Molli P, Urso P, et al. Tombstone Transformation Functions for Ensuring Consistency in Collaborative Editing Systems[C]//Proc. of CollaborateCom'06. Atlanta, Georgia, USA: IEEE Computer Society, 2006.

[23] 邵斌. 高效的操作转换一致性维护方法研究[D]. 上海: 复旦大学, 2010.

[24] Prakash A, Knister M J. A Framework for Undoing Actions in Collaborative Systems[J]. ACM Transactions on Computer-Human Interaction, 1994, 1(4): 295-330.

[25] Dewan P, Choudhary R. Coupling the User Interfaces of a Multiuser Program[J]. ACM Transactions on Computer Human Interaction, 1995, 2(1): 1-39.

[26] Sun Chengzheng. Undo as Concurrent Inverse in Group Editors[J]. ACM Transactions on Computer-Human Interaction, 2002, 9(4): 309-361.

[27] 杨江明, 顾宁, 吴筱媛. 基于地址空间转换方法的Undo操作支持[J]. 通信学报, 2006, 27(3): 48-56.

[28] Shao Bin, Li Du, Gu Ning. An algorithm for Selective Undo of Any Operations in Collaborative Applications[C]//Proc. of GROUP'10. Sanibel Island, USA: [s. n.], 2010.

[29] Shao Bin, Li Du, Gu Ning. ABTS: A Transformation-based Consistency Control Algorithm for Wide-area Collaborative Applications[C]//Proc. of CollaborateCom'09. Washington D. C., USA: IEEE Computer Society, 2009.

[30] Shao Bin, Li Du, Gu Ning. An Optimized String Transformation Algorithm for Real-time Group Editors[C]//Proc. of ICPADS'09. Washington D. C., USA: IEEE Computer Society, 2009.

[31] Xia Steven, Sun David, Sun Chengzheng, et al. Leveraging Single-user Applications for Multi-user Collaboration: The Coword Approach[C]//Proc. of CSCW'04. New York, USA: ACM Press, 2004.

[32] Sun Chengzheng, Xia Steven, Sun David, et al. Transparent Adaptation of Single-user Applications for Multi-user Real-time Collaboration[J]. ACM Transactions on Computer-Human Interaction, 2006, 13(4): 531-582.

[33] Welcome to CoPowerPoint: A Real-time Collaborative Multimedia Slides Creation and Presentation System[Z]. [2010-05-11]. http://cooffice.ntu.edu.sg/copowerpoint/.

[34] Zheng Yang, Shen Haifeng, Sun Chengzheng. Leveraging Single-user AutoCAD for Collaboration by Transparent Adaptation[C]//Proc. of CSCWD'09. Santiago, Chile: [s. n.], 2009.

[35] Agustina A, Liu Fei, Xia Steven, et al. CoMaya: Incorporating Advanced Collaboration Capabilities into 3D Digital Media Design Tools[C]//Proc. of CSCW'08. San Diego, USA: [s. n.], 2008.

[36] CoEclipse[Z]. [2010-06-21]. http://www3.ntu.edu.sg/home/czsun/projects/coeclipse/.

[37] Yang Jiangming, Wang Haixun, Gu Ning, et al. Lock-free Consistency Control for Web2.0 Applications[C]//Proc. of WWW'08. New York, USA: ACM Press, 2008.

[38] Shao Bin, Li Du, Gu Ning. A Sequence Transformation Algorithm for Supporting Cooperative Work on Mobile Devices[C]//Proc. of CSCW'10. Savannah, USA: [s. n.], 2010.

编辑　张　帆