# 协同编辑中的迟加入问题和协作可靠性

窦万峰[1,2], 王维丽[1,2]
(1. 南京师范大学数学与计算机科学学院, 江苏南京 210097; 2. 南京大学计算机软件新技术国家重点实验室, 江苏南京 210093)

摘 要: 实时协同编辑中协作站点在不同时机加入时, 由于网络延时或故障会导致其初始状态的不一致, 即同步问题. 同时, 协作过程中站点或网络故障或长时间无操作生成, 以及会话管理器的容错等问题会影响系统可靠性. 本文提出一个多迟加入 server 算法, 解决了迟加入 client 初始状态的一致性; 提出一种基于会话管理器的站点查询算法和分布式备份模型, 保证系统可靠协作. 通过实例对相关算法的有效性进行了验证.

关键词: CSCW; 协同编辑系统; 迟加入; 会话管理器; 分布式备份模型
中图分类号: TP391.7 文献标识码: A 文章编号: 0372-2112 (2005) 07-1275-04

# Problem of Late Join and Cooperation Reliability in Real-Time Cooperative Editing Systems

DOU Wan-feng[1,2], WANG Wei-li[1,2]
(1. *College of Mathematics and Computer Science, Nanjing Normal University, Nanjing, Jiangsu 210097, China;*
2. *State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210093, China*)

**Abstract:** Due to the network prolonging or fault, inconsistency or synchronous problem of the initial state of cooperative sites occurs when they join a real-time cooperative editing system at different time. Furthermore, the fault of cooperative sites or network, or long time no operations in cooperative sites, and the fault-tolerant of session manager can affect reliability of the system. A multiple later-join server algorithm is presented. The algorithm can effectively solve the consistency of the initial state of the late-join client. A query algorithm and distributed backup model based on session manager are proposed to guarantee the reliability of cooperative editing systems. The effectiveness of relative algorithms is verified by a concrete example.

**Key words:** CSCW; cooperation editing system; later-join; session manager; distributed backup model

## 1 引言

计算机支持的协同工作 (Computer Supported Cooperative Work, 简称 CSCW) 致力于并发控制、群组安全与可靠通信和高效的协作等研究. 自 20 世纪 80 年代中期提出发展至今, 越来越受到人们的重视. 协同编辑系统支持不同地域的协作者通过网络来共同高效地完成一个设计或编辑任务, 大大提高群组协作的效率[1~3]. 协作过程是一个有多人参与的过程, 通常这些参与者在地理位置上是分布的. 为了支持自由、自然的交互、可靠性的要求, 协同应用通常采用全复制式的体系结构, 即在每个协作站点具有共享对象的一个副本. 用户可以借助于这个本地对象完成操作, 然后再将操作传输给其他协作用户. 这种复制式结构带来的最大问题在于维护并确保分布数据的一致性的机制比较复杂. 然而在复制式体系结构下进行有效的初始化状态同步是系统必须解决的一个重要问题, 目前有关这个问题的解决策略还不是很多. 另外, 复制式环境下, 各协作站点的可靠协作和会话管理器的容错性等也是协同编辑系统实现中的一个重要问题之一.

## 2 协同用户的加入问题

采用集中式结构, 如加锁方法, 在编辑对象之前先给对象加锁, 以避免其他用户同时修改对象, 这需要一个集中式服务器来协调锁的申请和释放, 导致系统的网络瓶颈, 效率低、健壮性差. 复制式结构允许本地操作立即执行, 然后提交到其他站点, 因而响应速度快, 支持自由、自然的协作要求[1~3], 得到广泛研究. 但在复制式结构下, 由于网络延迟或故障等原因有可能导致各个协作站点上的同步数据不一致或同步数据不完整的问题.

对于一个实时协同编辑系统, 由于用户加入的时机不同, 他们操作的虽然是同一个共享对象, 但其初始状态有可能是不同的. 当一个新的参与者加入到某个实时协同编辑系统时, 首先检查共享对象是否打开, 若已打开则向会话服务器发送判断所有在线协作用户的共享对象是否为空的请求. 若共享对象为空, 则表示尚未有协作用户提交操作, 新加入者可以简单的做如下操作达到初始状态的同步: (1) 如果新加入者为初始站点, 即第一个协作者, 则创建一个会话并初始化成员列表

收稿日期: 2004-11-30; 修回日期: 2005-03-20
基金项目: 江苏省高校自然科学基金 (No. 02KJD520006; No. 03KJA520066; No. 04KJD520106)

和当前对象状态(即协作的共享文档);(2)如果新加入者为非初始站点,则向会话发起者站点发送请求进行共享对象同步.

当一个新的参与者加入到一个协同工作中,或用户再次加入时,它的初始状态与其他用户的就不同,就产生了应用初始状态同步或在原来状态基础上的状态同步,称为"迟加入"问题.称迟加入的站点为"迟加入 client",其他能为"迟加入 client"提供状态初始化的站点称为"迟加入 server".

### 2.1 单迟加入算法

传统的迟加入算法为单迟加入 server 算法,通过"最快响应法"选择"迟加入 server".当"迟加入 client"以组播方式发出迟加入同步请求后,将第一个做出应答的协同工作站点作为"迟加入 server"[4,5].对于消息响应较快的站点,通常工作状态比较良好,有利于提高迟加入站点状态同步的效率.但在复制式体系结构下,单迟加入 server 算法有可能导致同步数据不一致或不完整的问题.

如图 1 所示,在 $t0$ 时刻协作站点 x 已经发出一个数据包,此时"迟加入 client"尚未进入系统,该数据包在 $t4$ 到达"迟加入 server",而在 $t4$ 前,$t2-t3$ 这段时间内"迟加入 client"的状态同步已经完成,这样就造成了同步数据不完整的情况.

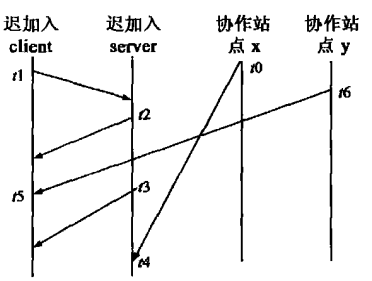


图 1 单迟加入 server策略存在的问题

另外,假设 $t2$ 时刻迟加入server 收到迟加入 client 状态同步请求后,响应同步请求并发送同步数据给迟加入 client,在此同步过程中,如果某在线站点 $y$ 在 t6 时刻提交操作,并在迟加入 client 同步结束 $t3$ 前把操作传送到迟加入站点,从而导致了协作站点共享对象不一致.

### 2.2 改进的多迟加入 server 算法

为了避免以上两种问题的发生,本文在多迟加入 server 算法[4]基础上提出了一种改进的多迟加入 server 策略.在多迟加入算法中并没有彻底解决同步数据不一致与不完整问题.改进的多迟加入 server 算法描述如下:

算法 1:用户迟加入控制算法

输入:*sid*:待加入的协作站点标识;*Sr*:加入会话 *r* 的所有用户;*r* 表示会话;

Begin

1. 判定 *sid* 是否为第一个协同站点,

(1)如果 *sid* 为第一个协同站点,则调用 CreateSession(*sid*, *r*, *Φ*)创建并初始化共享对象并返回;

(2)否则,判定所有在线站点的操作链表为空.

a)若为空,则调用 InitialState(*sid*)初始化自身并加入会话 *r*,返回;

b)否则,转下一步;

2. 向当前的每个站点发送暂停操作命令,得到响应的站点集合 *ss*;

3. 向每个响应的站点发送同步信息请求并得到同步数据:

Syn data ← RetriLastState(*ss*, *sid*);

4. 加入会话并启动协作过程.

End//结束

改进的多迟加入 server 算法通过发送暂停命令确保在同步过程中在线协作站点不会对共享对象做任何的操作,由所有在线站点把各自的操作序列分别发送给"迟加入 client",可避免单迟加入服务器算法可能带来的同步数据不一致、不完整的问题,同时多个站点传递同步数据改善了系统的性能,提高了效率.下面是读取同步数据的算法:

算法 2:RetriLastState(*ss*, *sid*)

1. 若会话中不再有成员,即|*ss*| = 0,则加入者初始化并创建会话,并返回;

2. 否则,向 *ss* 中每个成员站点 *s* 发送同步数据请求;

3. 循环读取每个站点 *s* 的同步数据;

4. 合并每个站点 syn data s 的同类信息;

5. 返回

该算法不是随机选取一个服务器站点,而是读取所有站点的同步数据.注意,读取每个协作站点的同步数据可能是冗余的,需要进行冗余数据合并.关于合并问题在下一节讨论.

### 2.3 进一步讨论

上面的多迟加入 server 算法在效率上存在两个问题,一是要从所有协作服务器上读取不同数据(一般是文档上所有已经执行过的操作)需要较长的时间;二是从各个服务器上读取的数据存在大量冗余的情况,需要合并.

为解决上述问题,设想从各个"迟加入服务器"上读取必需的操作,减少传输的数据量,提高系统效率,同时也可降低合并算法的复杂度.复制式环境下,本地操作立即执行然后传播到其他协作站点,如果直接读取服务器上"迟加入 client"没有的操作,可减少传输的数据量.在协同编辑中,为了比较操作的关系,引入了逻辑时间向量来记录操作之间的时间关系[6].

实时协同编辑系统中的操作是并发的,可以使用逻辑向量获取操作间的因果/并发关系[6].时间域由一组非负整数的 $N$ 维向量来表示($N$ 表示协作节点的数目).每个节点 $i$ 上都维护一个向量 $V[1, \cdots, n]$,$V[i]$ 是节点 $i$ 的本地逻辑时间,$V[j]$ 表示节点 $i$ 所知道的节点 $j$ 的最新本地时间.例如,如果 $V[j] = x$,节点 $i$ 知道节点 $j$ 广播时刻的逻辑时间为 $x$.整个向量 $V$ 表现了节点 $i$ 的全局逻辑时间,节点 $i$ 依据它为本地操作打上时间戳.节点上的逻辑时间向量随操作的进行不断更新,逻辑时间向量更新规则如下:(1)执行一个操作前,节点 $i$ 更新自己的逻辑时钟:$V[i] = V[i] + d$,$d$ 为增量,一般为 1;(2)每一个操作消息都附带了远程节点发送时刻的时间戳.当节点接收到操作消息($m$, $Vt$)时,执行以下操作:$N \geqslant k \geqslant 1$,$V[k] = \max(V[k], Vt[k])$;然后执行步骤(1).可以看出,本地站点维护的逻辑向量代表本地站点的最新状态.

首先,新加入客户随机选择一个站点服务器或从会话创建者上读取同步数据连同该站点的状态(即逻辑时间向量),

然后向其他站点发送同步数据请求时带上加入客户此时的最新逻辑向量, 服务器站点根据最新逻辑向量只发送该逻辑向量以后本地生成的操作. 改进同步数据算法如下:

算法 2: RetriLastState( ss, sid)

(1) 若会话中不再有成员, 即| ss| = 0, 则加入者初始化并创建会话, 并返回;

(2) 否则, 随机选取站点 s, 向 s 发送同步数据请求, 并读取同步数据 orisynr data;

(3) 从 orisynr data 中得到同步站点 s 的当前的逻辑时间向量 v;

(4) 向其他的站点 sk 发送带有最新逻辑时间向量 v 的同步数据请求;

(5) 循环获取同步期间其他站点 sk 生成的操作数据 synr dadarsk;

(6) 循环将其他站点 sk 的数据 synr dadarsk 加入到 orisynr data 中;

(7) 返回 orisynr data.

显然, 改进的算法使得合并更简单.

## 3　协作站点的可靠性

协作站点的可靠性和安全性包括两个方面的问题, 一是协作站点故障; 二是协作站点长时间没有操作信息. 协作站点故障会影响整个会话的效率甚至导致协作停顿, 因为其他协作站点在不知情的情况下始终给故障站点发送操作信息而浪费资源, 如果操作信息必须可靠传输, 那么协作站点必须等待确认, 导致协作停顿. 协作站点长时间没有操作信息, 尽管协作站点没有发生故障, 协作者可能离机没有退出会话, 这势必带来安全隐患(有可能信息被别人看到或者进行恶意的操作).

对于协作站点故障问题, 系统应能及时发现并从会话中清除该站点, 提高协作可靠性. 对于协作站点长时间无操作的情况, 系统也应能及时发现并强行将该站点的权限置为只读, 以防止恶意操作发生. 同时发出安全询问, 若无应答则清除该站点, 保证安全协作. 在会话管理器上实现一个组查询线程, 监视协作站点的运行情况. 会话管理器查询算法利用站点的逻辑时间向量来判定该站点是否有操作产生, 算法如下:

算法 3: GroupQuery( )

1. 延时器延时计算查询间隔;

2. 若超时, 循环向每个站点 s 发送查询, 即 SendQuery( s); 若站点无响应, 则清除该站点, 即 $Sr \leftarrow Sr - \{s\}$;

3. 循环向每个站点 s 发送操作信息查询, 即 SendQueryOp( s);

4. 循环读取每个站点 s 的最新状态 v, 即 ReceiveReqOp( s, v), 然后比较该站点当前操作状态和服务器记录的该站点最新操作状态:

(1) 若站点无操作产生, 即: $v[s] \leqslant vt[s]$, 则更改该站点权限为只读并通知其他站点;

(2) 否则更新服务器上该站点操作状, 即 $vt[s] = v[s]$.

## 4　会话管理器的容错性

会话管理负责协作组成员的信息维护、定期查询服务, 以及用户动态加入、退出等服务. 会话管理的一个问题就是容错, 如果会话管理器发生故障, 那么整个协作过程不得不停止, 影响了协作的可靠性. 文[7] 采用了主备份模型(primary-backup model). 两个会话管理器可以驻留在不同的站点, 并相互监听, 一旦主会话管理器故障, 备份管理器会立即取而代之, 维护协作过程.

主备份模型并不能在本质上解决可靠性. 从复制式结构分析来看, 会话管理器承担的任务不是很繁重, 而且服务器上维护的组成员信息在每个协作站点都有. 如果每个协作站点都作为主服务器的备份服务器可大大提高系统的可靠性. 本文给出一个分布式备份模型, 其原理是让每个在线协作成员充当会话管理器的备份, 当主服务器发生故障时, 协作成员任意一个会自动替代主服务器来维护会话信息和成员动态加入退出任务以及定期查询工作.

在分布式备份模型中, 每个站点都能接替主服务器的工作必然带来冲突问题, 因此需要提供一个控制机制. 采用这样的策略: 按照协作站点加入到会话的先后次序排序, 最前面的站点作为第一个备份机, 并与主服务器通信, 监听其状态. 一旦主服务器发生故障, 第一个站点接替服务器的工作, 同时第二个站点则成为备份机, 依次类推..., 从而保证系统可靠运行. 会话管理器在有站点加入或退出时进行重新排序, 并告知其他协作站点.

## 5　实例分析

在局域网环境下, 通过原型系统 CoDraftPaint 验证了上述算法的有效性. 三台主机 X、Y 和 Z, X 主机上的用户创建一个协同会话(如图 2a)所示), 并邀请 Y、Z 加入会话. Y 用户加入会话(没有任何一个站点提交操作), 其初始状态同 X. X、Y 分别提交了一个操作后(X 主机上的用户绘制了一条直线, Y 主机上的用户放大了圆), Z 加入会话. 在测试时随机产生一个网络延时, 则有可能当 Z 加入会话时, X、Y 本地操作已执行而远程操作由于网络延时还没有执行, 如图 2b) 所示. 传统的迟加入算法还可能出现 Z 状态同步结束前有某个协作站点提交了操作. 以上两种情况将会导致同步信息的不完整和不一致. 改进的多迟加入 server 算法解决了 Z 的初始状态同步问题(如图 2c) 所示).

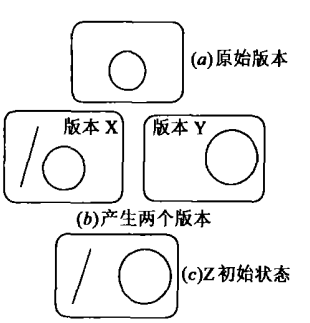


图 2　一个实例分析

## 6　相关工作比较和结论

目前解决迟加入问题的方法有单迟加入 server 和多迟加入 server 两种方法. 单迟加入 server 选择第一个响应的站点作为同步数据的来源, 其效率高. 但该方法有可能导致同步数据不一致或同步数据不完整的问题. 多迟加入 server 方法为解决单迟加入 server 存在的问题被提出. 存在的多迟加入 server 方法仅仅从各个站点读取数据, 而没有进行有效地合并冗余

数据,也没有考虑效率和可靠情况.本文的多迟加入 server 考虑上述不足,并采用逻辑时间向量从各个站点仅读取同步要求以后生成的操作.这些操作一般很少,同步效率高.同时,本文的方法只要得到第一站点的全部数据和其他站点少量的数据,因而网络负担也比较轻,效率自然高.

实时 CSCW 系统的可靠性和安全性是 CSCW 系统设计与实现中的一个重要的问题.本文主要针对实时协同编辑系统,深入讨论了系统中协作节点不同时机加入时,应用初始状态的同步问题,针对迟加入问题提出了一个改进的多迟加入 server 算法,该算法能够有效的解决迟加入 client 的共享数据的一致性问题.为了解决协作站点的可靠性和安全性,给出一个会话管理器查询算法,解决协作站点故障和长时间无操作信息的安全等问题.最后本文给出一个分布式备份模型解决了会话管理器的容错问题.本文提出的模型与算法已在原型系统 CoDraftPaint 进行了验证.进一步的工作是如何将上述的模型与方法与并发控制策略集成,为实时 CSCW 应用的开发提供支持.

参考文献:


[1] 窦万峰,李春萍.多版本中的对象标识及其压缩技术[J],软件学报,2004,15(8):1133-1140.
DOU Warr-Feng, LI Cherr-Ping. Object identification and its compression for multi-versioning technique[J]. Chinese J of Software, 2004, 15(8): 1133-1140(in Chinese).

[2] SUN C Z, et al. Achieving convergence, causality-preservation, and intentionr preservation in real-time cooperation editing systems[J]. ACM Transactions on Computer-human Interactions, 1998, 5(1): 63-108.

[3] YANG Guang-xin, SHI Mei-lin. Cova: A programming language for cooperative applications[J]. Science in China (Series F), 2001, 44(1): 73-80.

[4] 姜进磊,史美林.CSCW 中的对象同步与合并[J].计算机研究与发展,2003,40(9):1312-1318.
JIANG Jin-lei, SHI Mei-lin. Object synchronization and merging in CSCW[J]. Chinese Computer Research and Development, 2003, 40(9): 1312-1318(in Chinese).

[5] 姜波,卜佳俊.基于 Internet 的图像协同设计中迟加入问题的研究[J].计算机工程.2003,28(4):88-89.
JIANG Bo, PU Gui-jun. Research on the problem of late join in collaborative patter design on Internet[J]. Chinese J of Computer Engineering, 2003, 28(4): 88-89(in Chinese).

[6] Michel R, Mukesh S. Logical time: Capturing causality in distributed systems[J]. IEEE Computer Magazine, 1996, 29(4): 49-56.

[7] YANG Y, et al. Real-time cooperative editing on the internet[J]. IEEE Internet Computing, 2000, 4(3): 18-25.


作者简介:

窦万峰 男,1968 年 7 月出生于陕西永寿县,博士,副教授,主要研究领域为协同计算、软件工程.E-mail: douwf@email.njnu.edu.cn.

王维丽 女,1978 年 3 月出生于山东临沭县,硕士研究生,主要研究领域为计算机支持的协同工作.