基于 HLA所有权和 Agent组播的网络协作研究

汪红兵, 范植华, 佘春东
(中国科学院软件研究所通用软件实验室, 北京 100080)

摘 要: 介绍一种基于高层体系结构和多 Agent系统的网络协作框架, 给出网络协作必须满足的基本性质. 基于 HLA所有权和 Agent组播, 提出一种网络协作模型及算法 HOAM, 解决 HLA的实现软件 RTI目前不能支持消息因果顺序的不足. 证明算法满足因果约束. 比较分析了 LICRA和 HOAM算法的性能, 表明 HOAM算法作为一种简单的悲观算法也能达到与复杂的乐观算法 LICRA大约相同的性能.

关键词: 网络协作; 对象一致性; 高层体系结构; Agent组播
中图分类号: TP311.5 文献标识码: A 文章编号: 0372-2112 (2006) 05-0867-07

Research on Network Collaboration Based on HLA Ownership and Agent Multicast

WANG Hong-bing FAN Zhi-hua SHE Chun-dong
(General Software Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)

Abstract A network collaboration framework based on High Level Architecture and Multi-Agent System is introduced in this paper. Some required properties about network collaboration are presented. A network collaboration model and its algorithm HOAM are proposed based on HLA ownership and agent multicast. The shortage of being not satisfied with message causal order in the current implementation RTI of HLA is solved. Causality constraint supported by HOAM algorithm is proved. A comparative analysis is made in terms of the performance of HOAM and LICRA algorithm. It shows that HOAM, as a simple pessimistic algorithm, has the approximate same performance with LICRA, as a complex optimistic algorithm.

Key words network collaboration; object consistency; high level architecture; agent multicast

1 引言

高层体系结构 (High Level Architecture, HLA) 是一个通用仿真总线, 其目的是解决不同类型仿真应用之间的互操作及重用问题[1]. HLA包括联邦管理、对象管理、所有权管理、数据分发管理和时间管理 5种基础接口服务. HLA的实现软件 RTI已经成为一种通用的中间件, 不但支持各种仿真应用, 而且支持其他各种不同类型的应用, 如网络协作.

一般来说, 网络协作必须满足以下要求:

(1)因果约束 网络协作必须满足操作或消息的因果约束, 否则会产生不可理解的现象, 这是网络协作的基本要求.

(2)一致性 系统静止时刻共享对象一致性是网络协作的基本要求. 所谓静止时刻, 指的是系统中不存在任何消息传送的时刻. 网络协作领域, 一致性主要有两个标准: 二维 CSCW系统的 WYSIWIS和三维 CSCW系统的共享视点.

(3)快速响应 任一协作用户操作共享对象时应尽可能快地得到响应. 但是, 系统往往需要在因果约束、一致性和快速响应之间做出权衡.

(4)并发性 系统可同时支持并发用户数量.

(5)通信流量低 由于协作站点数目可能会很大, 因此, 低通信流量且其代价与协作站点数目无关的通信方式是可扩展的理想通信方式.

但是, 目前 RTI只支持两种基本的消息传递顺序: 接收顺序和时戳顺序. 一般仿真应用中, 消息传递顺序可分为四类: 接收顺序、优先级顺序、因果顺序和时戳顺序, 服务功能依次增强, 代价也随之增大. 接收顺序无法满足仿真的因果约束要求, 尤其是当网络不可靠或发生其他不确定性情况时. 而对于时戳顺序, 多个具有相同时戳顺序的

收稿日期: 2005-07-24; 修回日期: 2005-12-21

消息其接收顺序是不确定的, 同样无法满足因果约束要求, 且其代价最大. 因此, 需要对 RTI进行改造以支持消息因果顺序从而满足网络协作的要求, 而 RTI是一种商业软件, 用户进行源码的改造是不现实的.

传统的分布式计算和仿真领域, 出现了许多保证消息因果顺序的机制. 基于发生在前关系[2], 早期出现了 SES协议[3]. 此后, 针对 SES协议的消息具有 n^2 尺寸和仅支持广播的缺点, 一些研究者对此进行了改进, 主要有 MSES协议[4]. 另外, 在实现方面主要有嵌入式的因果顺序时间管理机制[5]和基于中间件的因果消息递交机制[6].

网络协作中, 由于客户端快速响应要求, 大多数网络协作使用乐观并发机制访问共享对象. 早期出现的是基于操作转换方法 dOPT[7], 算法为了保证用户良好的响应速度, 总是立即执行用户的本地操作. 此后, 一些研究者对该算法进行了改进, 主要有, 使用转换来对历史记录中的操作进行重新排序[8], 使用包含与排除变换使得进行操作转换的操作序列与操作具有一致的上下文[9]. 近年还有, 基于直接依赖关系的 LICRA 算法[10], 应用于 VRML网络编辑的 Mu3D协议[11], 基于树数据结构并结合仲裁模式的并发控制机制[12], 基于语义的并发控制框架[13], 集成 Agent组播和操作转换的方法[14], 实体为中心的并发操作控制机制[15].

但是, 乐观并发控制机制应用于网络协作也需要很大代价, 如搜索操作日志、进行操作转换以及恢复和撤销操作等. 当然, 如果网络协作应用于不可靠的网络通信环境中, 为了保证客户端快速响应要求, 需要使用乐观并发控制机制. 但是, 一般的可靠网络通信环境中, 如局域网, 使用悲观的并发访问机制, 尤其当各个协作任务的共享数据相关度较小时, 悲观机制由于其实现的简单往往表现出更大的适用性. 本文使用 HLA所有权管理服务实现共享对象的悲观并发控制.

基于 HLA所有权和 Agent组播, 提出一种网络协作模型及算法 HOAM (算法主要基于 HLA所有权管理和 Agent组播, 故命名算法为 HLA Ownership and AgentMulticast, 即 HOAM). 组织如下: 第二部分描述一个结合 RTI和 JADE的网络协作框架; 第三部分给出网络协作模型的相关定义; 第四部分介绍了 HOAM算法并对算法满足因果约束进行了证明; 第五部分比较分析 LICRA 和 HOAM 算法的平均响应时间; 最后, 第六部分对算法的相关特点进行总结并提出相关改进方向.

2 网络协作框架

2.1 框架描述

首先给出一种基于 RTI和 JADE的网络协作框架, 如图 1所示. RTI是 HLA的接口服务实现软件, JADE是一种遵从 FIPA规范的多 Agent系统开发平台. 框架基于 HLA联邦管理服务建立协作会话和 Agent组播组, 基于 HLA属性所有权管理服务建立对象属性级的锁机制, 基于 Agent组播支持网络协作的更新广播. 所谓协作会话, 指的是一次网络协作运行实例. 框架可同时支持多个协作会话. JADE和 RTI之间通过 RTI_JADE中间件进行连接, 中间件的功能包括 Agent对象管理、Agent交互管理和 Agent同步管理, 本文中主要实现将联邦管理服务的回调及时通知 Agent每个协作站点驻留一个管理 Agent, 代表其宿主站点参与网络协作.

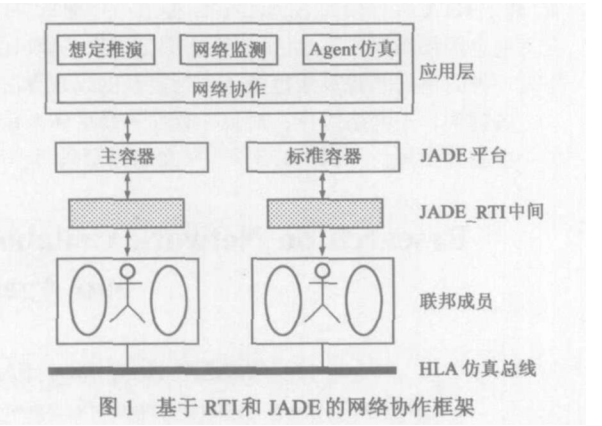


图 1 基于 RTI和 JADE的网络协作框架

另外, 框架还可以支持多 Agent系统仿真以及基于移动 Agent的网络监测等各种不同应用.

2.2 Agent组播

图 2给出了基于 JADE平台的组播实现. RMA、AMS以及 DF是 JADE中的三个基本的服务 Agent, 分别为远程监控 Agent, Agent管理服务和目录设施. 各个管理 Agent使用 AMS服务, 维护其组播组. 这里, 一个联邦成员对应一个网络协作客户端. 当系统中存在多个联邦时, 就存在相应的多个 Agent组播组. 图 2中给出了两个组播组: 即 Agent组播组包含 OP1、OPX1和 OPY1, 以及 Agent组播组包含 OP2、OPX2和 OPY2.

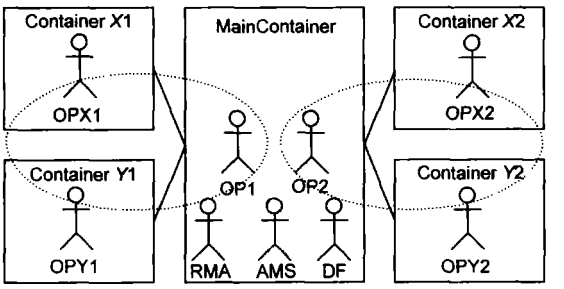


图 2 基于 JADE的 Agent组播

3 网络协作模型

3.1 对象

网络协作中, 对象分为两种: 私有对象和共享对象. 私有对象是各个协作站点内部使用的对象, 是协作站点的本地私有数据; 共享对象是各个协作站点公共操作的实体, 是协作运行的公共数据. 本文只关注共享对象.

定义 1　一个对象 OB 由对象标识、对象类型、父指针、孩子指针以及一系列属性组成 $\langle ObId, ObTp, Prt, \{Chd\}A_1, A_2, \cdots, A_m \rangle$, 其中, $ObId$ 是对象标识, $ObTp$ 是对象类型, Prt 是对象的父对象指针, $\{Chd\}$是对象的孩子对象指针, $A_i(i = 1, 2, \cdots, m)$是对象属性. 每个对象属性为一个 2元组 $\langle Type, Val \rangle$, 即属性类型和属性值.

对象类型 $ObTp$ 一般由具体应用语义确定. 例如, 虚拟装配网络协作中, 对象类型集合包含各种基本几何对象, 如柱体、球体和锥体等, 以及各种实际应用对象, 如发动机、线圈、电缆以及连接器等.

网络协作中, 由于对象的各个副本分布在不同站点上, 为维护各个副本之间的一致性, 需要进行对象标识. 本文使用两层编码进行对象标识. 一个对象的标识 $ObId$ 有两部分组成: $ObId = \langle SiteId, OlHId \rangle$, $SiteId$ 是对象所在的站点标识, $OlHId$ 是对象的本地层次编码, 如图 3 所示. 本地对象是按照树结构组织的, 因此, 根节点的 Prt 为空, 叶节点的$\{Chd\}$为空, 其他节点的父指针与孩子指针分别为指向树结构的父节点和孩子节点.

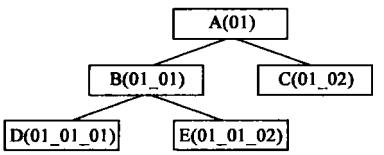


图 3　本地对象编码

3.2　权限访问矩阵

为了支持基于属性粒度的权限管理, 对象属性作了如下扩充, 每个对象除具有一般的属性 A_i 外, 还具有两个属性 pToCrt和 pToDlt 当客户需要在某个节点下创建新对象时, 必须获得该节点对象的 pToCrt属性所有权; 当客户需要删除某个节点对象时, 必须获得该节点对象的 pToDlt属性所有权(pToDlt实际为 HLA 的 privilegeToDelete属性). 因此, 每个站点需维护一个权限访问矩阵, 如表 1所示.

表 **1**　权限访问矩阵

	READ	UPDATE	CREATE	DELETE
$OB_1.A_1$	1	0	/	/
$OB_1.A_2$	0	1	/	/
……	1	0	/	/
$OB_1.A_m$	1	1	/	/
$OB_1.$ pToCrt	/	/	1	/
$OB_1.$ pToDlt	/	/	/	0

注: 1表示具有该权限; 0表示没有该权限; /表示无意义

3.3　操作及因果关系

本文中, 将操作分为本地操作和网络操作. 本地操作指的是读操作, 只发生在本地, 不需要对读消息进行广播; 网络操作包括创建、更新和删除操作, 虽发生在本地, 但需要对更新进行消息编码并进行广播. 由于读操作只发生在本地, 因果顺序所约束的操作不包括读操作.

定义 2　一个操作 OP 由操作标识、操作类型、操作参数组成 $\langle OpId, OpTp, OpPa \rangle$, 其中, $OpId = \langle SiteId, Seq \rangle$是操作标识, $OpTp = \{CREATE, READ, UPDATE, DELETE\}$是操作类型, $OpPa = \langle OB, \{OB.A_i, pToCrt, pToDlt\} \rangle$是操作参数.

对于操作标识 $OpId = \langle SiteId, Seq \rangle$, $SiteId$ 标识操作发生的站点, Seq为发生在站点上操作序列计数.

定义 3　两个操作 OP_i 和 OP_j, OP_i 因果优先于 OP_j, 记为 $OP_i \rightarrow OP_j$, 满足下面条件之一:

(1)OP_i 和 OP_j 执行于同一个站点, 且 OP_i 的执行先于 OP_j;

(2)OP_i 和 OP_j 执行于不同站点, 但执行 OP_j 的站点是在收到 OP_i 操作并递交后才执行 OP_j;

(3)存在某个操作 OP_k, 使得满足 $OP_i \rightarrow OP_k \wedge OP_k \rightarrow OP_j$.

定义 4　两个操作 OP_i 和 OP_j, 满足 $OP_i \rightarrow OP_j$, 若不存在 OP_k, 使得满足 $OP_i \rightarrow OP_k \wedge OP_k \rightarrow OP_j$, 则称操作 OP_i 是操作 OP_j 的直接前因.

3.4　消息

传统网络协作中, 使用基于时间向量机制来保存因果关系, 消息具有 $O(n)$复杂度 (n为协作站点个数). 因此, 基于时间向量的网络协作算法扩展性差. 这里, 给出一种基于 2个标量的消息机制, 具有良好的扩展性.

定义 5　一个消息由更新操作及其直接前因操作的标识组成 $m = \langle OP, OpIdPre \rangle$, 其中, OP 为更新操作, $OpIdPre$ 是 OP的直接前因操作标识.

3.5　管理 Agent

网络协作模型中, $Agent$作为管理实体, 代表 $Agent$所在的协作站点参与网络协作管理. 一个协作站点只存在一个管理 $Agent$, 各个管理 $Agent$使用组播进行通信.

定义 6　一个管理 $Agent$是一个 8元组 $Agent_i = \langle H_i, O_i, W_i, \{L_i^k\}, S_i, OpId_i, OAM_i, G_i \rangle$, 其中:

(1)H_i为 $Agent_i$中已经执行的操作列表;

(2)O_i为 $Agent_i$中对象属性所有权列表;

(3)W_i为 $Agent_i$中被等操作列表;

(4)$\{L_i^k\}$为 $Agent_i$中已达到但由于直接前因操作还没有到达的等待操作列表;

(5)S_i为 $Agent_i$的操作序列号;

(6)$OpId_i$为 $Agent_i$的当前已执行操作标识;

(7)OAM_i为 $Agent_i$的当前对象属性权限访问矩阵;

(8)G_i为 $Agent_i$的组播列表;

3.6　网络协作模型

网络协作模型中, 每个协作站点中均驻留一个管理 Agent代表其参与网络协作管理.

定义 7　一个网络协作模型是一个 3元组 $CoApp = \langle Site, Agent, L \rangle$, 其中, $Site = \{Site_i\}$是参与协作的协作站点集合, $Agent = \{Agent_i\}$是参与网络协作的管理 $Agent$集合, $L: Site \leftrightarrow Agent$是协作站点集合到管理 $Agent$集合的一一对应映射, 它反应了管理 $Agent$的驻留位置关系. 本文中, 有 $L(Site_i) \leftrightarrow Agent_i$.

4 算法

4.1 算法描述

根据网络协作模型, 基于 HLA 所有权和 Agent组播, 提出算法 HOAM, 描述如下:

$Agent_i$ 站点上:

/* 初始化阶段 */

$H_i = \Phi$; //本地已经执行的操作列表, 初始为空

$O_i = \Phi$; //本地对象属性所有权列表, 初始为空

$W_i = \Phi$; //本地被等操作列表, 初始为空, 每个
//被等操作对应一个本地等待操作列表

$\{L_i^k\}$; //本地等待操作列表, 每个列表中操作的存
//放顺序按照定义 4给出的直接前因关系
//所确定的顺序

$S_i = 0$; //本地维护的操作序列号

$OpId = (0, 0)$; //本地维护的当前操作标识

OAM_i; //本地维护的访问权限矩阵

G_i; //本地维护的 $Agent$组播组成员列表

T_i; //本地维护的可接受的最大响应时间

/* $Agent$组播组维护阶段 */

(1)若 Fed_i 创建联邦, 则 $Agent_i$ 创建组播组 G_i;

(2)若 Fed_i 删除联邦, 则 $Agent_i$ 删除组播组 G_i;

(3)若 Fed_i 加入联邦, 则 $Agent_i$ 加入组播组 G_i;

(4)若 Fed_i 退出联邦, 则 $Agent_i$ 退出组播组 G_i;

/* 执行阶段 */

/* 产生本地操作 OP */

Genarate OP;

OP. OpId = (i, S_i + 1);

Case CREATE: //本地创建操作

OAM_i 中查找 OP. OB. Prt pToCrt访问权限, 若有, 则继续; 若无, 则返回; O_i 中查找 OP. OB. Prt pToCrt所有权, 若有, 则继续; 若无, 则请求所有权转移 OP. OB. Prt pToCrt, 进入等待(最大延迟为 T_i); Exec (OP, L);

Case UPDATE: //本地更新操作

OAM_i 中查找 OP. OB. A_k 访问权限, 若有, 则继续; 若无, 则返回; O_i 中查找 OP. OB. A_k 所有权, 若有, 则继续; 若无, 则请求所有权转移 OP. OB. A_k, 进入等待 (最大延迟为 T_i); Exec (OP, L);

Case DELETE: //本地删除操作

OAM_i 中查找 OP. OB. pToDlt访问权限, 若有, 则继续; 若无, 则返回; O_i 中查找 OP. OB. pToDlt所有权, 若有, 则继续; 若无, 则请求所有权转移 OP. OB. pToDlt, 进入等待 (最大延迟为 T_i); Exec (OP, L);

Case READ: //本地读操作

OAM_i 中查找 OP. OB. A_k 访问权限, 若有, 则继续; 若无, 则返回; 返回 OP. OB. A_k 值;

/* 收到操作组播消息 */

Receive m = (OP, OpIdPre)

(1)若 OpIdPre∈ {H, OP. OpId}, 则 Exec (OP, R); 否则转本部分 (3);

(2)若 OP. OpId∈ W_i, 则设 L_i^k 为等待 OP的操作列表, 执行 Exec($\{OP \in L_i^k\}$, R) 和 RemoveFromList(OP. OpId, W_i);

(3)若 OpIdPre∈ $\{L_i^m$. OP. OpId$\}$, 则 AddToList(OP, L_i^m); 否则转本部分 (5);

(4)若 OP. OpId∈ W, 设 L_i^k 为等待 OP的列表, 执行合并操作列表 Converge(L_i^m, L_i^k)和 RemoveFromList(OP. OpId, W_i);

(5)执行 AddToList(OpIdPre, W_i), 并创建等待 OpIdPre 的操作列表 L_i^k, 并执行 AddToList(OP, L_i^k);

/* 收到所有权请求回调 */

ReceiveRequest (A= {A_k, pToCrt, pToDlt})

(1)验证 $A \in O_i$ 和 CompletedWithOP(A), 即属性是否还在本地列表且属性相关操作已经完成. 若是, 则继续; 若否, 则返回;

(2) RemoveFromList(A, O_i), 并调用 RTI服务通知所有权释放;

/* 获得所有权释放回调 */

ReceiveResponse (A= {A_k, pToCrt, pToDlt})

(1)执行 AddToList(A, O_i);

(2)本地等待所有权循环终止, 执行操作;

/* 执行 CREATE/UPDATE/DELETE操作子过程 */

Exec(OP, F= {L, R}) //L: 本地操作; R: 远程操作

Case L:

CASE CREATE: //创建对象实例

SceneExec OP;

registerObjectInstance(OP. OB);

AddToList({OB, A_k}, O_i);

CASE DELETE: //删除对象实例

SceneExec OP;

deleteObjectInstance(OP. OB);

RemoveFromList({OB, A_k}, O_i);

CASE UPDATE: //更新对象实例

SceneExec OP;

AddToList(OP, H_i);

m= (OP, (i, S_i));

AgentiMulticast(m, G_i);

$S_i = S_i + 1$;

OpId= (i, S_i);

Case R:

SceneExec OP;

AddToList(OP, H_i);

OpId= OP. OpId;

算法主要分为三个阶段: 初始化阶段, Agent组播组维

护阶段和执行阶段. 初始化阶段完成本地数据结构的初始化; Agent组播组维护阶段使用 HLA 联邦管理服务实现协作会话支持和维护 Agent组播组; 执行阶段最终实现操作的网络组播, 支持网络协作.

执行阶段可以分为产生本地操作、收到操作组播消息、收到所有权请求回调和获得所有权释放回调四个部分. 其中, 产生本地操作部分, 所有读操作均在本地完成, 所有更新操作 (包括创建、修改和删除) 都需要进行更新广播. 每次更新操作之前, 需要获取相关对象属性的所有权, 而拥有属性所有权的管理 Agent只有在确认与该属性的相关更新操作完成之后, 才会释放属性所有权. 每个管理 Agent还维护一个本地最大响应时间 T_i 以及相关的计时器. 从开始获取所有权开始计时, 当时间延迟 T_i 后, 若仍没有获得该实例所有权, 则返回"超时"消息, 并取消本地操作; 收到操作组播消息部分, 基于定义 4的操作直接前因关系, 维护本地等待操作列表 $\{L_i^k\}$和本地被等操作列表 W_i, 并当收到被等操作消息后, 执行列表删除和合并, 并执行远程操作 (算法中使用 Exec(OP, F= {L, R}) 子过程来统一表示更新操作. 当 F= L时表示执行本地操作; 当 F= R 时表示执行来自于其他站点的远程操作); 收到所有权请求回调和获得所有权释放回调两个部分实现对象属性所有权的维护.

另外, 算法对 HLA 对象实例和一般对象实例进行区分. HLA 对象实例只是为了管理属性所有权而存在, 一般对象实例是为了具体应用而存在. 算法中使用 SceneExec 表示一般对象的操作, 使用注册对象实例操作 (registerObjectInstance)和删除对象实例操作 (deleteObjectInstance)维护 HLA对象实例.

综上所述, HOAM 算法使用 HLA 联邦管理服务实现协作会话支持和维护 Agent组播组, 使用 Agent所有权管理服务维护对象属性锁, 更新操作消息发布通过 Agent组播来完成. 一次联邦运行相当于一次协作会话, 同时建立一个相应的 Agent组播组. 由于 HLA 可以同时支持多个联邦运行, 因此, 算法可以同时支持多个协作会话运行.

4.2 因果约束证明

首先给出关于算法满足因果约束的分析. 算法中对于 $\{L_i^k\}$每个等待操作列表的操作存放顺序是按照定义 4给出的直接前因关系来进行的. $\{L_i^k\}$的每个等待操作列表对应一个 W_i 中的操作. 具体实现时, 使用指针进行关联, 即 W_i 中每个操作指向一个 L_i^k 等待操作列表. 由于算法执行操作总是按照 $\{L_i^k\}$所确定的直接前因关系所确定的顺序来进行, 所以算法能保证遵从因果约束.

算法的目的是满足网络协作的各个站点的管理 Agent能够按照消息因果约束所确定的操作顺序处理各种操作, 保证各个管理 Agent具有相同的操作处理顺序, 从而构成一致的网络协作.

假设有两个操作 OP_0, OP_n 具有如下关系:

(1) $OP_0 \rightarrow OP_n$;

(2) OP_0, OP_n 都提交给 $Agent_i$ 处理.

因此, 证明算法满足操作因果约束, 即证明: 对于任意 $Agent_i$, $Agent_i$ 执行 OP_0 在 OP_n 之前.

首先, 如果 OP_0 是 OP_n 的直接前因操作, 即不存在 OP_k, 使得 $OP_0 \rightarrow OP_k \wedge OP_k \rightarrow OP_n$, 则由算法的操作提交过程可以得出, OP_n 提交处理必须等到 OP_0 提交处理之后. 因此, $Agent_i$ 执行 OP_0 在 OP_n 之前.

其次, 如果 OP_0 不是 OP_n 的直接前因操作, 则 $\exists OP_i$ (i = 1, 2, …, n- 1), 满足如下关系:

$$OP_0 \rightarrow OP_1 \rightarrow \cdots OP_i \cdots \rightarrow OP_n$$

根据数学归纳法, 假设操作集合 $\{OP_0, OP_1, \cdots, OP_{n-1}\}$是按照操作的因果顺序进行提交. 由于 OP_{n-1}是 OP_n 的直接前因操作, 则 OP_n 提交处理必须等到 OP_{n-1}提交处理之后. 而操作集合 $\{OP_0, OP_1, \cdots, OP_{n-1}\}$是按照操作的因果顺序进行提交的, OP_{n-2}先于 OP_{n-1}, OP_{n-3}先于 OP_{n-2}, …, OP_{i-1}先于 OP_i (i= 1, 2, …, n - 1), …, 因此, 根据递推关系和归纳假设, $\{OP_0, OP_1, \cdots, OP_n\}$是按照操作的因果顺序进行提交. 因此, $Agent_i$ 执行 OP_0 在 OP_n 之前.

因为 $Agent_i$ 是任意的, 故算法满足因果约束得到证明.

4.3 算法分析

HOAM 算法与 dARB及其改进类算法相比, HOAM 算法的更新操作消息具有 $O(2)$的复杂度, 比之使用时间向量消息的 $O(n)$ (n 为协作站点个数)复杂度, 算法的网络消息是常量级的, 具有良好的扩展性.

HOAM 算法与 Mu3D协议相比, HOAM 算法的锁粒度为对象属性级, 而 Mu3D 的锁粒度为 VRML 的节点及其孩子节点组成的子树, 因此, HOAM 算法发生所有权转移的概率更小, 算法支持的网络协作的并发度更高.

实际应用中可以使用基于空间的划分方法对协作环境中的共享对象进行划分, 使得各个协作站点具有最少的数据相关度, 从而减少所有权转移的次数.

5 性能分析

前面给出了算法满足因果约束的证明. 但是, HOAM 算法作为一种悲观并发访问控制机制, 平均响应时间作为一个衡量网络协作性能的指标, 显得更为重要. 这里, 对乐观算法 LICRA和悲观算法 HOAM 的平均响应时间进行比较分析.

为了实验的简单性, 只以更新操作为研究对象. 实验环境如下: 联想 P4 2 6GHz/256M RAM 计算机 6台, 通过 100M 网卡互连组建成可靠通信的局域网环境. 协作站点均匀地分布于各个计算机. 实验参数: 30个共享对象按树数据结构进行组织, 每个对象平均具有 10个属性.

由图 4可以看出: (1) HOAM 算法在 10个协作站点参与协作时, 客户端平均响应时间大约为 180ms 这基本能满足一般的网络协作要求; (2) HOAM 算法和 LICRA 算法具

有大约相同的平均响应时间. 这是因为, LICRA 算法虽然是一种乐观并发访问机制, 但操作每次执行前都需要与存储在操作日志中的操作进行比较, 并且当发生冲突时, 还需要进行操作转换, 而 HOAM 算法虽然是一种悲观机制, 但其锁机制是建立在 RTI的高效的通信和所有权管理机制上, 且锁层次是对象属性级, 因此, 就平均响应时间而言, 悲观算法 HOAM 可以达到与乐观算法 LICRA 大约相同的性能.

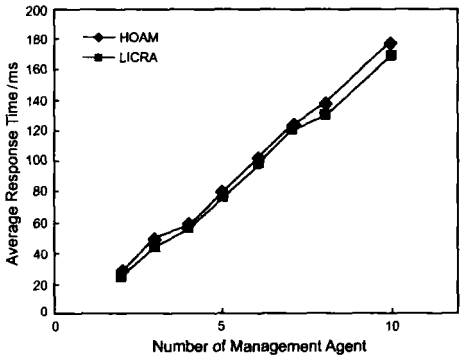


图 4 HOAM 和 LICRA 算法的平均响应时间比较

6 结论

基于 HLA 所有权和 Agent组播, 提出一种网络协作模型及其算法 HOAM. 具有以下几个特点:

(1)算法基于 HLA 所有权和 Agent组播, 本质上是一种悲观算法, 实现简单, 但能达到与乐观算法 LICRA 大约相同的性能.

(2)算法解决了目前 HLA 实现软件 RTI不能支持消息因果顺序的不足.

(3)算法的更新消息具有 $O(2)$复杂度, 克服传统的基于时间向量的网络协作算法更新消息的 $O(n)$复杂度的缺点, 具有良好的扩展性.

(4)算法基于 HLA 所有权管理服务, 实现网络协作中共享对象一致性的维护.

(5)算法集成了属性级别的权限管理, 且锁机制位于对象属性层次, 增加了网络协作的并发度.

(6)算法基于 HLA 联邦管理服务, 支持协作会话和维护 Agent组播组, 能支持管理 Agent选择性地动态加入和退出特定的协作会话.

(7)算法基于 Agent参与网络协作管理, 为以后基于 Agent的协作建模及实现和基于 Agent的兴趣度管理的集成奠定了基础.

算法的一些改进方向有: (1)当有多个协作站点(或联邦成员)同时请求同一个属性所有权时, 算法没有提供选择决策机制; (2)实现基于 Agent的兴趣度管理和协作环境对象划分, 使得各个协作站点具有最小的数据相关度, 从而减少所有权转移的次数, 是下一步需要解决的问题.

参考文献:

[1] DMSO. High Level Architecture Rules, Interface Specification, Object Model Template Version 1.3 [OL]. http://www.dmso.mil. 1998.

[2] Lamport L. Time, clocks and the ordering of events in a distributed system [J]. Communications of the ACM. 1978, 21(7): 558-565.

[3] Schiper A, Eggli J, Sandoz A. A new algorithm to implement causal ordering[A]. Proc. of International Workshop on Distributed Algorithms[C]. LNCS. Springer-Verger 1989. 392. 219-232.

[4] Cai W, Lee B S, Zhou J. Causal order delivery in a multicast environment: an improved algorithm [J]. Journal of Parallel and Distributed Computing. 2002, 62: 111-131.

[5] Lee B S, Cai W, Zhou J. A causality based time management mechanism for federated simulation[A]. Proc. of 15th Workshop on Parallel and Distributed Simulation (PADS 2001)[C]. Washington, IEEE Computer Society. 2001. 83-90.

[6] Turner S J, Cai W, Chen J. A middleware approach to causal order delivery in distributed simulations[A]. Proc. Of the 2003 European Simulation Interoperability Workshop (Euro-SIW)[C]. Stockholm, Simulation Interoperability Standards Organization. 2003, No. 03E-SIW-085.

[7] Ellis C A, Gibbs S J. Concurrency control in groupware systems[A]. Proc. of the ACM SIGMOD Conf. on Management of Data[C]. Seattle, ACM Press. 1989. 399-407.

[8] Suleiman M, Cart M, Ferrie J. Serialization of concurrent operations in a distributed collaborative environment[A]. Proc. of the ACM SIGGROUP Conf. on Supporting Group Work[C]. Phoenix, ACM Press. 1997. 435-445.

[9] Sun C Z, Ellis C. Operational transformation in real-time group editors: issues, algorithms and achievements[A]. Proc. of the ACM Conf. on Computer Supported Cooperative Work[C]. Seattle, ACM Press. 1998. 59-68.

[10] Kanawati R. LICRA: A replicated data management algorithm for distributed synchronous groupware applications [J]. Parallel Computing Journal. 1997, 22: 1733-1746.

[11] Ricardo G, Yuhua L. Mu3D: a causal consistency protocol for a collaborative VRML editor[A]. Proc. of the Fifth Symposium on Virtual Reality Modeling Language (Web3D-VRML)[C]. New York, ACM Press. 2000. 53-62.

[12] Ionescu M, Marsic I. Tree-based concurrency control in distributed groupware[J]. Computer Supported Cooperative Work. 2003, 12(3): 329-350.

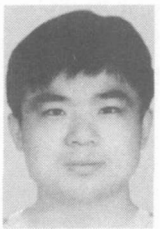
[13] Yang G X, Shi M L. oodOPT: A semantics-based concurren-

cy control framework for fully-replicated architecture[J]. Journal of Computer Science and Technology, 2001, 16(6): 531-534.

[14] 杨武勇, 史美林, 姜进雷. 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): 497-503

[15] 余春艳, 候宏仑, 吴明晖, 潘云鹤. 协同设计中以设计为中心的并发操作控制机制[J]. 计算机辅助设计与图形学学报. 2004, 16(9): 1289-1294.

作者简介:



汪红兵　男, 1978年出生于安徽巢湖, 博士研究生, 主要研究领域为多 Agent系统、分布式计算、计算机仿真.
E-mail: wawahohobb@163.com

佘春东　男, 1971年出生, 副研究员, 硕士生导师, 主要研究领域为并行计算、数据挖掘、智能 Agent. E-mail: shrcd@sina.com

范植华　男, 1942年出生, 研究员, 博士生导师, 主要研究领域为 Agent技术、并行化、数字化仿真. E-mail: fan_zhihua@hotmail.com