RTI 中乐观推进机制的实现*

刘步权+, 王怀民, 姚益平

(国防科学技术大学 计算机学院,湖南 长沙 410073)

Implementation of Optimistic Advancing Mechanism in RTI

LIU Bu-Quan+, WANG Huai-Min, YAO Yi-Ping

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573697, E-mail: bqliu@nudt.edu.cn, http://www.nudt.edu.cn

Received 2003-11-07; Accepted 2003-11-27

Liu BQ, Wang HM, Yao YP. Implementation of optimistic advancing mechanism in RTI. *Journal of Software*, 2004,15(3):338~347.

http://www.jos.org.cn/1000-9825/15/338.htm

Abstract: In distributed modeling and simulation area, it is intractable to comprehend the fundamental principles of optimistic advancing mechanism and implement the optimistic advancing services in RTI (runtime infrastructure) according to HLA (high level architecture) specifications. This paper introduces two different optimistic advancing mechanisms in PDES (parallel discrete event simulation) and HLA, and reveals some important differences between them. For example, the virtual time can be rolled back in PDES, the logical time can not be rolled back in HLA, and an optimistic federate can only roll back its message-scheduling time and must ensure that its rollback won't influence the advancing of conservative federates. Rollback occurs in a logical process in PDES, but it can only occur in a federate rather than RTI in HLA. In addition, a new implementation mechanism called Zero-Saving is proposed in this paper. With this mechanism, a RTI does not need to save any execution states when optimistic advancing services are implemented. This mechanism has successfully been applied to a RTI named StarLink. The Zero-Saving mechanism adds two new variables into the message retraction handle type RTI::MessageRetraction Handle which is a data type defined by IEEE 1516.1. One variable represents the time stamp of the sent TSO (time stamp order) message, and the other is used to save all federates which receive the message. When a TSO message is sent to RTI, RTI returns the sending federate a message retraction handle with all message-received federates. So RTI knows which federates should be notified to retract the received messages whenever the sending federate uses a message retraction handle to ask RTI to retract a TSO message. The fundamental principles and implementation of optimistic advancing services introduced in this paper are useful for RTI developers.

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2001AA115127 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.G1999032703 (国家重点基础研究发展规划(973))

作者简介: 刘步权(1969－),男,江苏姜堰人,博士生,讲师,主要研究领域为分布式仿真,分布对象技术,操作系统;王怀民(1962－),男,博士,教授,博士生导师,主要研究领域为分布计算技术,信息安全;姚益平(1963－),男,博士,研究员,主要研究领域为分布式仿真,虚拟现实.

Key words: parallel discrete event simulation; high level architecture; time management; optimistic advancing mechanism; zero-saving

摘 要: 正确理解乐观推进机制的基本原理,并按照高层体系结构 HLA(high level architecture)规范实现 RTI(runtime infrastructure)中的乐观推进服务一直是分布式仿真领域关注的难点问题.介绍了并行离散事件仿真 PDES(parallel discrete event simulation)和 HLA 中的乐观推进机制,并指出了它们之间的重要差异,例如 PDES 中的虚拟时间(virtual time)可以回卷(roll back),回卷发生在进程中;而 HLA 中的逻辑时间不能够回卷,但乐观盟员在不影响保守盟员推进的情况下可以回卷自己调度事件的时间,回卷发生在盟员内而不是 RTI 内.另外,提出了在实现乐观推进服务时,不需要 RTI 作任何保存操作的"零保存"技术,并成功地将该技术应用到 RTI 软件 StarLink 中."零保存"技术通过在消息句柄类型 RTI::MessageRetractionHandle 定义中增加两个变量,一个表示 TSO(time stamp order)消息的时标,另一个表示所有接收该消息的盟员(RTI::MessageRetractionHandle 为 IEEE 1516.1 定义的数据类型),RTI 将具有此类型的消息句柄返回给发送 TSO 消息的盟员保存,当发送盟员再次使用消息句柄撤消(retract)消息时,RTI 从消息句柄中就可以知道并通知接收盟员撤消消息.对于理解和开发 RTI 中的乐观推进服务具有重要的现实意义.

关键词: 并行离散事件仿真;高层体系结构;时间管理;乐观推进机制;零保存

中图法分类号: TP391 **文献标识码**: A

为了保证仿真的正确执行和消息之间的因果关系,高层体系结构 HLA[1](high level architecture)引入了时间管理服务[2].时间管理服务的本质是通过逻辑时间对分布式仿真中的消息进行排序,因此时间管理也可以理解为"时序管理".在并行离散事件仿真 PDES[3](parallel discrete event simulation)的基础上,HLA 提出了保守时间推进和乐观时间推进两种方式,并支持这两种方式的混合推进.

保守推进机制要求 RTI(runtime infrastructure)必须严格保证盟员(federate)按照时标序大小接收消息,因此,盟员在推进的过程中永远不会产生冲突.然而,即使两个消息的时标不同,但是它们之间却未必存在因果关系;如果两个连续的消息不存在因果关系,却不一定需要按照时标序大小来调度它们.因此,保守推进机制以"时标序"来确定"因果序"是一种过度"保守"的消息处理机制.鉴于保守推进机制的不足,HLA 引入了乐观推进机制.在乐观推进机制下,盟员总是"乐观地"认为下一个消息的到来不会与当前的系统状态产生冲突,然而这种"赌博"不可避免地会存在一定的风险,风险的程度依赖于特定的应用.一旦出现冲突,乐观盟员需要通过"回卷"(rollback)机制来解决这种冲突.乐观机制在处理器的设计、操作系统中的页面调度、分布式数据库中的事务处理以及并行编译等领域都有很重要的应用.

20 多年来,国内外一直关注乐观推进技术的研究.但国外大多在 PDES 领域进行研究,即使在 HLA 标准发布之后,我们也很少看到针对 HLA 乐观推进机制的研究文章.在 PDES 领域,研究的是纯乐观推进技术,与保守推进技术基本不相关;而在 HLA 领域,却存在保守推进机制和乐观推进机制的协同问题.自从 HLA 标准公布之后,国内的众多科研单位就展开了 HLA 技术的研究和 RTI 软件的开发.时间管理服务一直是 RTI 开发的重点和难点,国内在这方面的研究也比较多[4~8].但就乐观推进机制而言,除了文献[4,5]的作者结合自己的实际工作在学术方面做了较好的研究以外,我们尚没有看到其他更深入的文章,特别是在 RTI 中如何实现乐观推进服务方面.而理解 HLA 规范并在 RTI 中实现乐观推进服务正是本文所要研究和解决的问题.

StarLink 是我们在国家"863"计划资助下,按照 IEEE 1516 系列标准[1]并基于 CORBA[9]中间件技术研制成功的 RTI 软件.为了更好地说明 RTI 中乐观推进服务的实现技术,本文以集中式体系结构的 StarLink 为例来说明(StarLink 的另一版本为基于多个 RTI 服务器互连的层次式体系结构[10,11]).在集中式体系结构下,所有盟员的状态信息(包括 TSO 队列、RO 队列)都保存在中央 RTI 服务器,底层消息的传输由 CORBA 中间件自动解决并对 StarLink 设计者透明,盟员由于没有 LRC(local RTI component)而不能够实现相互间的直接通信,因此一个盟员发送的 TSO 消息必须通过中央 RTI 服务器才能够发送到接收盟员.集中式体系结构对于设计者按照 HLA 标准实现 RTI 以及初学者理解 HLA 规范都要比其他 RTI 体系结构方便和简单.

我们认为,乐观推进机制的实现需要建立在正确理解 HLA 规范的基础上,但是对于 HLA 规范的理解不能够仅停留在规范本身,而需要结合 PDES 中的乐观推进机制进行.然而,HLA 中的乐观推进机制虽然来源于 PDES,但与 PDES 中的乐观推进机制相比却存在巨大的差异.本文在分析和比较了两种乐观推进机制的差异之后,提出了不需要 RTI 作任何保存工作的"零保存(zero-saving)"机制,并将其成功地应用到 StarLink 软件的实现中.

1 两种乐观机制的比较

就我们开发 RTI 的经验可知,能否正确区分 PDES 和 HLA 中的乐观推进机制对于实现 RTI 乐观推进服务至关重要.在这一节,我们主要讨论下列内容:PDES 中的乐观推进机制、HLA 中的乐观推进机制、两种推进机制的比较、RTI 中的消息重构问题、HLA 仿真中用户如何使用乐观推进机制进行程序设计.

1.1 PDES中的乐观机制

Jefferson 首次提出了基于 Time Warp 机制[12]的乐观推进算法,在文献[12]中提出的许多概念一直为以后的乐观推进算法所沿用,文献[5]对其基本原理进行了很好的介绍.本文将结合一个实例来说明 PDES 中的乐观推进机制,更详细的讨论可参见文献[13].

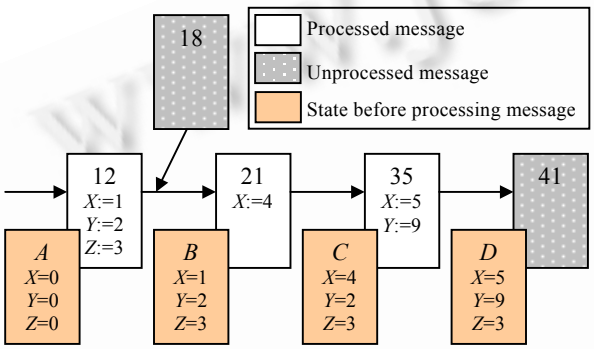

Fig.1 Process state of straggler message arriving
图 1 "过时"消息到达时的进程状态

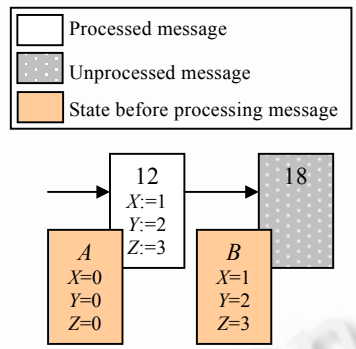

Fig.2 Process state after rollback
图 2 回卷后的进程状态

图 1 描述了进程"冒进"处理消息的过程(PDES 中的"进程"类似于 HLA 中的"盟员").在图 1 中,一个进程的状态用 3 个变量 X,Y 和 Z 表示.进程起始状态 A 为 X=0,Y=0,Z=0;处理完时标为 12 的消息(简称消息 12)后,进程状态 B 为 X=1,Y=2,Z=3;消息 21 只改变 X 的内容,该消息处理后的进程状态 C 为 X=4,Y=2,Z=3;处理完消息 35 后,进程状态 D 为 X=5,Y=9,Z=3;此时将要处理下一个消息 41,但是因为该进程收到一个其他进程发来的较小消息 18,因此进程必须先处理消息 18.我们看到一个有趣的现象,如果消息 18 只是改变 Z 的值,而消息 21 和 35 都没有改变 Z 的值,因此在处理完消息 35 后,不需要回卷状态就可以"安全"地处理消息 18;如果消息 18 只是改变 Y 的值,而消息 21 并没有改变 Y 的值,因此进程可以回卷到状态 C,再继续处理消息 18;同样,如果消息 18 需要改变 X 的值,则进程需要回卷到状态 B,图 2 描述了这样的状态,此时队列中的下一个待处理消息为 18.

对于 PDES 中的乐观推进机制,我们获得了如下一些有用的结论:

命题 1. PDES 中的虚拟时间可以回卷.

在 PDES 中,保守机制使用逻辑时间[14]的概念,而乐观机制采用虚拟时间[12]的概念,本质上都是对仿真时间的抽象表示,只是习惯用法不同而已.在 HLA 中,由 RTI 统一负责时间管理;但在 PDES 中却没有进行时间管理的中心机构,时间管理和时间推进均由进程负责,当发生冲突时可以回卷进程的虚拟时间.

命题 2. PDES 中的进程只能撤消(retract)自己发送给其他进程的消息.

图 1 中进程处理到消息 18 时需要回卷状态,假设回卷到状态 B.此时进程只需要撤消在消息 21 和 35 处理期间发送给其他进程的消息,消息 21 和 35 是否撤消取决于其他进程的行为,但不能够由该进程撤消.以消息 21

为例说明,假设本进程为 P,发送消息 21 的进程为 Q.进程 P 处理完消息 18 后,给进程 Q 发送了一个消息 m,与前面的论述相同,m 可能导致进程 Q 发生冲突而撤消消息 21,也可能因为没有发生冲突而不会撤消消息 21.如果消息 21 没有被撤消,那么消息 21 仍然保留在进程 P 的消息队列中.

1.2 HLA中的乐观机制

按照 IEEE 1516 系列标准,HLA 中的乐观机制由下列 3 个服务组成,前两个为盟员调用 RTI 的服务,由 RTI 负责实现;后一个为 RTI 回调盟员的服务,由盟员负责实现.

(1) 盟员请求 RTI 推进服务 flushQueueRequest,简记为 FQR;

(2) 盟员请求 RTI 撤消消息服务 retract;

(3) RTI 请求盟员撤消消息服务 requestRetraction.

HLA 乐观推进机制的大致过程为:首先由盟员通过 FQR 服务向 RTI 请求消息,RTI 将该盟员 TSO 消息队列中的所有消息发送给该盟员;盟员将接收到的消息存放到程序的消息队列中并按照时标序大小依次处理消息(参见第 1.5 节),当出现冲突时,调用 RTI 的 retract 服务请求撤消自己发送的 TSO 消息;RTI 收到 retract 服务时,如果消息已经发送给其他盟员,则向其他盟员发送 requestRetraction 回调服务.

对于 HLA 中的乐观推进机制,我们获得了如下一些有用的结论:

命题 3. HLA 中的盟员调度时间可以回卷.

为了论述方便,本文引进术语“调度时间(message-scheduling time)”.调度时间是指盟员处理一个消息时该消息所具有的时标.在 PDES 中,当进程处理一个消息时,该进程就推进到该消息的时标处.因此,在 HLA 的盟员中,我们也认为当盟员处理一个消息时,该盟员的调度时间就推进到该消息的时标处.通俗地讲,调度时间指的是“什么时候的事就什么时候调度”.对于保守盟员而言,调度时间≤当前逻辑时间;对于乐观盟员而言,调度时间可以≤当前逻辑时间(不会发生回卷),但调度时间通常≥当前逻辑时间(有可能导致回卷).

命题 4. HLA 中的逻辑时间不能够回卷.

盟员和 RTI 都不能够回卷逻辑时间,这里所说的逻辑时间是指任意时刻盟员的当前逻辑时间,即可以通过 queryLogicalTime 服务调用的时间.对于盟员而言,盟员的逻辑时间是由 RTI 控制的,盟员只能够请求 RTI 向前推进逻辑时间,如果请求推进的逻辑时间<盟员的当前逻辑时间,则 RTI 不会同意盟员的请求并引发 RTI::LogicalTimeAlreadyPassed 异常.对于 RTI 而言,如果允许回卷乐观盟员的逻辑时间,那么乐观盟员就不可能与保守盟员“和平共处”.

保守机制和乐观机制的“和平共处”还可以从 retract 服务的前提条件看出,IEEE 1516.1 标准规定 retract 服务只能够撤消那些时标值>(T+Lookahead)的消息(如果盟员处于 Grant 状态,那么 T 表示盟员的当前逻辑时间;如果盟员处于请求推进状态,那么 T 表示盟员请求推进的逻辑时间).另外,标准还规定 retract 服务只能由校准盟员(regulating federate)调用,因此保守盟员的最大可用逻辑时间 GALT[8](greatest available logical time)一定≤回卷盟员的(T+Lookahead),这样,乐观盟员想要撤消的消息一定是保守盟员还没有收到的消息,保证了两类盟员能够“和平共处”. IEEE 1516 系列标准中的最大可用逻辑时间 GALT 与 HLA 1.3 中的 LBTS[15](lower bound time stamp)意义相同.

命题 5. HLA 中约束盟员(constrained federate)的逻辑时间一定≤GALT.

首先对保守盟员而言,该命题显然成立,否则该盟员是不安全的.其次由 IEEE 1516.1 标准可知:盟员调用 FQR(Trequest)请求推进到 Trequest 时,RTI 允许该盟员推进到 Tgrant:

$$T\text{grant}=\min\{\text{minTSO},\text{GALT},T\text{request}\}.$$

这里的 minTSO 表示 TSO 队列中的最小消息的时标.因此,Tgrant≤GALT.盟员使用乐观推进服务 FQR 推进时,其逻辑时间总是安全的.

本命题讨论的是约束盟员的情形.对于非约束盟员而言,盟员只能接收 RO 消息而无法接收 TSO 消息,因此非约束盟员使用时间推进服务时意义不大,GALT 的取值依赖于特定的 RTI 软件.

命题 6. HLA 中的回卷发生在盟员内而不是 RTI 内.

IEEE 1516 系列标准是关于互操作平台 RTI 的标准,而不是上层应用即盟员的标准.在标准中自始至终都没有出现“roll back”或“rollback”等表示“回卷”的词汇,而是用“retract”表示盟员请求 RTI 撤消一个已经发送的消息;至于是否发送 retract 服务则取决于盟员.

命题 7. HLA 中的盟员只能撤消自己发送给 RTI 的消息.

该命题本身就是 retract 服务的前提条件之一.

1.3 推进机制的比较

在前面讨论的基础上,两种乐观推进机制的比较可归纳为表 1.

Table 1 Comparison of two optimistic mechanisms

表 1 两种乐观机制的比较

Optimistic mechanism	PDES	HLA
Time management	Controlled by logical process	Controlled by RTI
Time advancing	Controlled by logical process	Federate requests RTI to advance
Rollback	Occur in logical process	Occur in federate, not RTI
	Virtual time can be rolled back	Message-Scheduling time can be rolled back; logical time can't be rolled back
Message retraction	A logical process or federate can only retract messages sent by itself	
	Send anti-message[12]	Send message retraction handle
Relation with conservative mechanism	Irrelated	Peacefully coexist

1.4 两个有争议的问题

要实现 RTI,还必须搞清两个问题,即是否允许 RTI 通知接收盟员撤消 RO 消息?是否允许 RTI 进行消息重构?由于 IEEE 1516 系列标准并没有明确地说明,因此我们认为 RTI 赞成和不赞成这两个观点都是可以的,但一定要给出一个明确的答复,因为对这两个问题的回答与 RTI 中数据结构的设计以及 TSO 消息队列的维护直接相关.下面两个命题表明了 StarLink 的观点.

命题 8. RTI 应当通知接收盟员撤消 RO 消息.

该命题的意思是说,一个乐观盟员调用 retract 服务撤消一个消息 m 时,消息 m 可能被多个定购盟员所接收;在这些盟员中有些是以 TSO 方式接收的,有些是以 RO 方式接收的;在消息 m 被撤消时,RTI 显然要调用 requestRetraction 服务通知那些以 TSO 方式接收的盟员,但 RTI 是否要通知那些以 RO 方式接收的盟员呢?StarLink 支持这一命题,因为 requestRetraction 服务是由盟员方实现的,如果盟员不需要处理这样的情形,那么它可以不实现该服务,RTI 的通知就无效;如果盟员实现该服务,则表明它支持这一结论,RTI 的通知就有效.因此支持这一命题具有一定的灵活性.

命题 9. RTI 不负责乐观盟员的消息重构.

以图 1 为例,假设图 1 是基于 HLA 的乐观推进机制.当盟员处理完消息 35 后,收到了小时标的消息 18,需要回卷到状态 B.此时盟员可以采取两种方式:

(1) 清除盟员消息队列中保存在消息 18 之后的所有消息(注意,这里的消息队列并不是保存在 RTI 中的 TSO 消息队列).

(2) 不清除盟员消息队列中保存在消息 18 之后的所有消息,只有收到 RTI 发送的 requestRetraction 服务时才从队列中删除相应的消息.

对于第 1 种情形,既增加了 RTI 设计的复杂性,又增加了盟员程序的复杂性,降低了盟员程序的效率.与 PDES 中介绍的情形类似,如果处理完消息 18 后,消息 21 没有被发送者撤消,那么该盟员下一个要处理的消息就是消息 21,但按照方法(1),当队列清空后怎样才能再次获得消息 21 呢?这就需要 RTI 来保存该消息,增加了 RTI 设计的复杂性.

第 2 种情形实际上借用了 PDES 所使用的方法,由盟员自己建立并维护消息队列,RTI 在收到来自盟员的 FQR 请求时,一次性地将 TSO 队列中的所有消息发送到盟员并清空 TSO 队列,盟员将接收到的消息存放到自己的队列中.这种方法对于 RTI 和盟员来说都很方便.本文提出的“零保存”机制必须建立在该命题的基础之上.

1.5 盟员程序如何应用乐观推进机制

由于 HLA 将时间管理机制与上层应用即盟员分开,通常我们讨论更多的是如何按照标准实现 RTI.但是用户应该如何使用乐观推进机制设计应用程序呢?本文建议使用下列算法:

(1) 建立一个接收消息队列 *A*.该队列用来保存来自 RTI 的所有消息,包括 TSO 消息和 RO 消息.

(2) 建立一个消息句柄队列 *B*.该队列用来记录所有状态下发送给 RTI 的 TSO 消息的消息句柄,供撤消消息时使用.

(3) 建立一个状态队列 *C*.该队列用来保存程序的状态空间.

(4) 设置一个后台线程(或者使用 Windows 系统下的定时器).后台线程每隔一定的时间间隔保存一次程序状态(注意:不需要每处理一个消息就保存一次程序状态,否则保存操作会导致过多的存储开销,也会降低程序的执行效率).假设盟员的当前逻辑时间为 T,则$<T$ 的状态只保存一个,即离 T 最近的状态;$<T$ 的其他状态则被删除.因为盟员采用 FQR 请求推进时,RTI 总是同意其推进,并把当前逻辑时间返回给该盟员.采用 T 回收状态空间要比采用 GALT 要强,因为后者需要不断地调用 queryGALT 服务来查询 GALT 值,增加了不必要的开销.另外,后台线程在回收状态空间的同时还需要清理队列 *A* 和 *B* 中与被回收状态空间对应的所有不需要保存的队列元素.

(5) 盟员使用 FQR 向 RTI 请求消息,将收到的消息存放到消息队列 *A* 中,并依次处理各个消息.

(6) 当盟员收到 requestRetraction 回调服务时,从消息队列中查找该消息,若该消息尚未被处理则从队列中删除;否则,判断是否需要回卷程序状态.

(7) 若需要回卷程序状态,则向 RTI 发送 retract 服务撤消所有在“冒进”状态下发送的消息,这些消息的句柄被保存在队列 *B* 中.

(8) 最后回卷程序状态.回卷可能是个连锁反应的过程.

2 乐观机制的实现

在前面讨论的基础上,本节给出集中式 StarLink 乐观推进机制的实现方法.正如前面所言,RTI 只需要实现 FQR 和 retract 两个服务.本节主要讨论下列内容:管道机制和可靠性、RTI 服务器的总体结构、FQR 服务的实现和 retract 服务的实现.在讨论 retract 服务时,本文给出一种称为“零保存”的实现方法,该方法不需要 RTI 保存任何状态. “零保存”机制同样适用于类似 DMSO RTI 体系结构[14]的分布式 RTI 软件的开发.

2.1 管道机制和可靠性

管道机制是指,在 RTI 和盟员之间传输消息时遵循“先发送的消息最先被接收到”的原则.对于时间管理服务而言,管道机制尤为重要.如果 RTI 软件在实现时间管理服务时不遵循管道机制,那么无论是盟员调用 RTI 还是 RTI 调用盟员时都会导致消息的混乱和时序不一致现象.实现时间管理服务的另外一个要求是传输的可靠性.例如,对于保守盟员而言,如果推进请求被网络丢弃了,则该盟员会因为等待 RTI 的同意而处于无限等待状态,其他保守盟员也会因为该盟员无法推进而处于等待状态,这样就导致仿真无法继续进行.

TCP 传输协议能够有效地解决这两个问题,而 UDP 传输协议则难以解决.StarLink 是在国产 CORBA 中间件 StarBus[16]的基础上实现的,StarBus 在实现时只采用了 TCP 协议而没有混杂 UDP 协议.虽然 StarBus 为上层应用程序提供了可靠和不可靠两种消息传输方式,但是这两种方式都是基于 TCP 协议实现的.StarLink 调用 StarBus 来实现消息在分布式系统中的传输,传输过程对 StarLink 开发者透明,StarLink 调用 StarBus 与盟员调用 RTI 服务类似.

2.2 RTI的总体结构

如图 3 所示,集中式 StarLink 实现时的总体结构由 3 个大的类对象组成.RTIambassador 对象包含多个 Federate 对象,每个 Federate 对象包含一个 CallbackThread 对象.每个对象的作用如下:

(1) RTIambassador 对象.该对象包含一个 FederateList 类型的变量,用来指向所有 Federate 对象实例的存储

空间.另外,还包含 RTI 提供给盟员调用的所有 HLA 服务,包括乐观推进机制的 FQR 和 retract 服务.RTIambassador 对象还包含实现时间管理服务时必不可少的两个私有函数:用于计算 GALT 的 countGALT 函数和用于推动其他盟员前进的 pushFederates 函数.

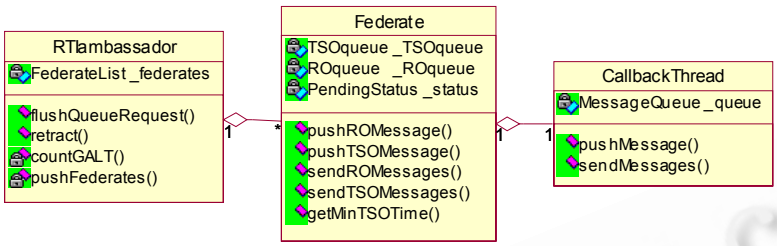

Fig.3 Data structure of StarLink with central RTI architecture

图 3 集中式 StarLink 的数据结构

(2) Federate 对象.该对象用来保存盟员的执行状态,图中包含 3 个变量:TSO 消息队列、RO 消息队列和表明盟员是否处于挂起状态的布尔变量.因为盟员在使用乐观推进服务 FQR 时不会被挂起,所以盟员只会因为使用保守推进服务时而被挂起.当盟员被挂起时,按照 HLA 标准规定,盟员在 RTI 同意其推进之前不能够再调用各类服务请求推进(包括 FQR).Federate 对象包含 5 个函数,两个 push 函数用于将 RO 和 TSO 消息分别放入到盟员的 RO 和 TSO 消息队列中,两个 send 函数用来从相应的队列中取出消息并将其放入到后台线程 CallbackThread 的队列中.getMinTSOTime 函数用来返回 TSO 消息队列中最小消息的时标.

(3) CallbackThread 对象.当盟员加入到 RTI 时,RTI 为每个盟员生成一个 Federate 对象实例,该实例自动创建一个后台线程用于将来自 RTI 的消息(包括 RO 和 TSO 消息)传输给盟员.由此可以看出,StarLink 采用的是一种基于多线程的并发传输技术.当 CallbackThread 收到来自 Federate 对象的消息时,一个信号量被触发,表明收到一个消息,于是立即将其发送到盟员.

2.3 推进服务的实现

推进服务 FQR 的函数原型为

void flushQueueRequest(RTI::LogicalTime theTime).

FQR 服务的实现算法如下:

(1) 处理异常.如果盟员没有加入联盟或者已经处于挂起状态(_status 变量为真),则引发异常;如果请求推进的逻辑时间 theTime 小于盟员的当前逻辑时间,则引发异常.

(2) 计算 GALT.盟员 GALT 的计算方法可参见文献[8],GALT 的计算与推进服务的类型、盟员所处的状态、Lookahead 以及 TSO 队列中的最小时标消息等因素相关;但绝不能简单地归结为 $\min\{T_i+L_i\}$(T_i 为盟员 i 的当前逻辑时间;L_i 为盟员 i 的 Lookahead).

(3) 重新计算当前逻辑时间 T.如果盟员不是约束盟员(constrained federate),则 T=theTime;否则 T=min{minTSO,GALT,theTime}.其中,theTime 为调用服务的参数,minTSO 为通过 getMinTSOTime 函数获得的盟员 TSO 队列中最小消息的时标.

(4) 处理 Lookahead.如果调用 modifyLookahead 服务降低盟员的 Lookahead,则 Lookahead 只能在推进的过程中逐渐变小.此时必须检查盟员是否处于这样的一种状态,并重新计算 Lookahead.

(5) 发送 RO 消息.调用 Federate 对象的 sendROMessages 函数,将所有 RO 消息放入到后台线程的消息队列中,后台线程立即将其发送到盟员.

(6) 发送 TSO 消息.调用 Federate 对象的 sendTSOMessages 函数,将所有 TSO 消息依时标序大小放入到后台线程的消息队列中,后台线程立即将其发送到盟员.

(7) 同意盟员推进.调用 timeAdvanceGrant 回调服务,允许盟员推进到逻辑时间 T.

(8) 推动其他盟员前进.如果盟员为校准盟员,则检查所有因推进请求而没有得到 RTI 同意的盟员,重新计算它们的 GALT,看能否因为本盟员的推进而使得它们获得前进.

2.4 撤消服务的实现

撤消服务的函数原型为

void retract(const RTI::MessageRetractionHandle& theHandle).

2.4.1 两个问题

要实现该服务必须解决好两个问题: “定购者”问题和“拆包”问题.

“定购者”问题.当 RTI 收到盟员的撤消消息服务 retract 时,RTI 是如何知道 theHandle 参数所表示的消息被哪些定购者接收了呢?一种方法是建立一个队列,队列中的每个元素是消息句柄到接收者集合的映射.然而要维护该队列将是复杂的,具体体现在:在盟员加入联盟时需要创建该队列;在盟员退出联盟时需要清除和摧毁该队列;在盟员发送 TSO 消息时需要加入新的元素;在盟员撤消消息时需要删除已有元素;在盟员前进时需要清理该队列;在盟员变为非校准盟员时也需要清理该队列等诸多情形.StarLink 提出的“零保存”方法可以有效地解决这一难题(该术语与操作系统中的“零拷贝 zero-copy”[17]术语类似).

“拆包”问题.按照 HLA 规范,乐观盟员 *A* 发送的一个 TSO 消息有可能被拆成 4 个消息发送给定购盟员 *B*.这 4 种消息类型为:可靠的 RO 消息、不可靠的 RO 消息、可靠的 TSO 消息和不可靠的 TSO 消息.按照命题 8,如果 RTI 在撤消 RO 消息时也需要通知盟员,那么假如 4 个消息全部保存在 RO 或 TSO 消息队列中(还没有发送到盟员 *B*),则将 4 个消息从消息队列中删除即可,就不需要回调 requestRetraction 服务通知盟员 *B*;但如果仅有部分消息保存在消息队列中,其他消息已经发送到盟员 *B*,那么除了清除保存在消息队列中的消息之外,还必须通知盟员 *B* 撤消其他消息.StarLink 对“拆包”问题进行了简化处理:一个对象只要有一个属性为可靠传输,则所有属性都采用可靠传输;否则采用不可靠传输.这样,当乐观盟员 *A* 发送的一个 TSO 消息在传输给定购盟员 *B* 时,最多只能被拆分成一个 RO 消息和一个 TSO 消息.

2.4.2 “零保存”机制

下面介绍“零保存”机制的基本原理.

(1) 在 StarLink 中,RTI::MessageRetractionHandle 类型用接口定义语言 IDL[18]描述如下:

```
struct MessageRetractionHandle {
  UniqueID          theSerialNumber;
  FederateHandle    sendingFederate;
  LogicalTime       theTime;
  FederateHandleSeq receivingFederates;
};
```

该类型包含 4 个变量:惟一标识一个消息的序数 theSerialNumber,发送 TSO 消息的盟员 sendingFederate,消息的时标 theTime,以及接收该消息的所有盟员集合 receivingFederates.与 DMSO 开发的 RTI 相比,这里的定义多了后两个变量.RTI::MessageRetractionHandle 类型对应于 HLA 1.3 中的 RTI::EventRetractionHandle 类型,IEEE 1516 系列标准严格区分了“event”和“message”这两个概念,前者为内部行为,后者为网络行为.

当盟员 *A* 发送了一个 TSO 消息时,RTI 必须将 RTI::MessageRetractionHandle 类型的消息句柄返回给发送盟员 *A*.消息句柄的设置方法为:theSerialNumber 为一个计数器,用来区分不同的 TSO 发送消息,每发送一个 TSO 消息,计数器加 1;sendingFederate 和 theTime 分别设置为发送盟员的句柄和消息的时标;如果一个句柄为 *h* 的盟员 *B* 接收了 TSO 消息,则将 *h* 加入到集合 receivingFederates 中;如果接收了 RO 消息,则将(1000000+*h*)加入到集合 receivingFederates 中.

(2) “零保存”机制的基本思想是:由消息句柄携带所有的接收盟员,并将消息句柄返回给发送者保存.这样,发送者在使用 retract 服务向 RTI 请求撤消消息时,RTI 根据消息句柄中保存的接收盟员集合就可以知道究竟有哪些盟员接收了该消息(RTI 能够区分 RO 消息和 TSO 消息).当然,该机制必须建立在命题 9 的基础之上.

2.4.3 retract 服务的实现

使用“零保存”机制实现 retract 服务的过程非常简单,算法如下:

(1) 处理异常.如果盟员没有加入联盟,盟员不是校准盟员,theHandle.SendingFederate 不等于调用 retract 服务的盟员,则引发异常(theHandle 为 retract 服务携带的消息句柄);如果 theHandle.theTime≤(*T*+Lookahead)也要引发异常,以保证不会撤消保守盟员已经收到的消息(如果盟员处于 Grant 状态,则 *T* 表示盟员的当前逻辑时间;如果盟员处于推进状态,则 *T* 表示请求推进的时间).这样就实现了乐观盟员和保守盟员的"和平共处".

(2) 令集合*Φ*为空,集合*Φ*用来保存所有应该通知撤消消息的盟员.依次处理 theHandle.receivingFederates 集合中的每个元素 *h*:

如果 *h*<1000000,则表明盟员 *h* 接收了 TSO 消息.如果 theHandle 所指定的消息在盟员 *h* 的 TSO 队列中,则删除此消息;否则将 *h* 加入到集合*Φ*.

如果 *h*≥1000000,则令 *h*=*h*−1000000,表明盟员 *h* 接收了 RO 消息.如果 theHandle 所指定的消息在盟员 *h* 的 RO 队列中,则删除此消息;否则将 *h* 加入到集合*Φ*.

(3) 依次向集合*Φ*中的所有盟员发送 requestRetraction(theHandle)回调服务.这里令参数 theHandle 中的 receivingFederates 集合为空,以节约网络带宽.注意,即使一个盟员同时接收了 TSO 和 RO 消息,也只需要通知一次即可.接收盟员通过 theHandle.theSerialNumber 匹配消息,以决定撤消消息和回卷操作.

对于类似 DMSO RTI 体系结构的分布式 RTI 而言,消息句柄 theHandle 从 LRC 返回给发送盟员时需要携带 receivingFederates 集合,在 LRC 之间传输时可令 receivingFederates 集合为空.

命题 10. HLA 中的保守盟员也能够撤消消息.

参照 retract 服务的前提条件可以知道:虽然保守盟员没有使用 FQR 服务推进,但仍然可以撤消自己发送的 TSO 消息.另外,一个程序可以同时采用包括 FQR 在内的多种时间推进服务,没有必要因为某个盟员采用了 FQR 服务推进逻辑时间就将其称之为乐观盟员,或者说使用了乐观推进机制;即使一个盟员采用了 FQR 服务,如果该盟员的 TSO 队列中的消息都是安全的,则该盟员一定不会因为本次推进请求而发生回卷.因此,我们不主张过分区分 HLA 中的保守推进机制和乐观推进机制,否则会与 HLA 能够适应各种类型仿真的"框架"思想相违背.

3 结束语

自从 HLA 标准发布之后,时间管理服务一直是仿真界关注的焦点问题之一.但基于 HLA 乐观推进机制的研究还不够深入,特别是在 RTI 中如何实现乐观推进服务,盟员如何使用乐观推进机制进行程序设计等问题都让 RTI 设计者和仿真应用开发者感到困惑.本文在深入研究 PDES 和 HLA 两种乐观推进机制的基础上,结合多年 RTI 开发经验,对两种不同的推进机制进行了分析和比较,指出了它们之间的差异,给出了仿真应用程序的开发方法,为基于 HLA 规范实现乐观推进服务提供了理论基础.另外,本文还按照 IEEE 1516 系列标准给出了乐观推进服务在集中式 StarLink 中的完整实现,提出了简化 RTI 设计的"零保存"机制,并成功地将其运用到 StarLink 中.实际上, "零保存"机制能够有效地应用于不同体系结构的 RTI 软件的开发.

References:

[1] Simulation Interoperability Standards Committee (SISC) of the IEEE Computer Society. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—IEEE Std 1516-2000, 1516.1-2000, 1516.2-2000. New York: Institute of Electrical and Electronics Engineers Inc., 2000.

[2] Fujimoto RM. Time management in the high level architecture. Simulation, 1998,71(6):388~400.

[3] Fujimoto RM. Parallel and distributed simulation systems. In: Peters BA, Smith JS, Medeiros DJ, Rohrer MW, eds. Proc. of the 33nd Conf. Winter Simulation. Washington: IEEE Computer Society, 2001. 147~157.

[4] Yao XY, Huang KD. Algorithms of real-time control and optimistic time synchronization based on time management in HLA. Journal of National University of Defense Technology, 1999,21(6):84~87 (in Chinese with English abstract).

[5] Ouyang LL, Song X, Qing DZ, Hao JB, Wang J. Research of time management in HLA and simulation algorithms of PDES. Journal of System Simulation, 2000,12(3):237~240 (in Chinese with English abstract).

[6] Zhang L, Yin WJ, Chai XD, Liu M. Investigation on time management of RTI. Journal of System Simulation, 2000,12(5):494~498 (in Chinese with English abstract).

[7] Huang J, Huang KD. Time management in the high level architecture. Computer Simulation, 2000,17(4):69~73 (in Chinese with English abstract).

[8] Liu BQ, Wang HM, Yao YP. A non-deadlock time management algorithm. Journal of Software, 2003,14(9):1515~1522 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1515.htm

[9] 2003. http://www.corba.org/

[10] Liu BQ, Wang HM, Yao YP. Key techniques of a hierarchical simulation runtime infrastructure--StarLink. Journal of Software, 2004,15(1): 9~16 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/9.htm

[11] Yao YP, Lu XC, Wang HM. Design and implementation of hierarchical RTI server. Chinese Journal of Computers, 2003,26(6): 716~721 (in Chinese with English abstract).

[12] Jefferson DR. Virtual time. ACM Trans. on Programming Languages and Systems, 1985,7(3):404~425.

[13] Fujimoto RM. Parallel and Distributed Simulation Systems. New York: John Wiley & Sons, Inc., 2000. 97~136.

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

[15] Department of Defense Modeling and Simulation Office. RTI 1.3-Next Generation Programmer's Guide Version 4. 2001. http://www.dmso.mil

[16] 2002. http://www.863.org.cn/863_95/indust/ind24.html

[17] 1996. http://citeseer.nj.nec.com/chu96zerocopy.html

[18] 2003. http://www.service-architecture.com/web-services/articles/omg_interface_definition_language_idl.html

附中文参考文献:

[4] 姚新宇,黄柯棣.基于 HLA 时间管理的实时时间控制和乐观时间同步算法设计.国防科学技术大学学报,1999,21(6):84~87.

[5] 欧阳伶俐,宋星,卿杜政,郝江波,王锦.HLA 时间管理与 PDES 仿真算法研究.系统仿真学报,2000,12(3):237~240.

[6] 张龙,尹文君,柴旭东,刘民.RTI 系统时间管理算法研究.系统仿真学报,2000,12(5):494~498.

[7] 黄健,黄柯棣.HLA 中的时间管理.计算机仿真,2000,17(4):69~73.

[8] 刘步权,王怀民,姚益平.一种无死锁的时间管理算法.软件学报,2003,14(9):1515~1522.http://www.jos.org.cn/1000-9825/14/1515.htm

[10] 刘步权,王怀民,姚益平.层次式仿真运行支撑环境 StarLink 中的关键技术.软件学报,2004,15(1):9~16.http://www.jos.org.cn/1000-9825/15/9.htm

[11] 姚益平,卢锡城,王怀民.层次式 RTI 服务器的设计与实现.计算机学报,2003,26(6):716~721.