RTI 时间管理的一种新型动态尺度标注算法

杜星玥[1a]，卢 昱[1b]，陈立云[1a]，党若雯[2]

（1. 军械工程学院 a. 计算机工程系；b. 训练部，石家庄 050003；2. 四川大学 电气信息学院，成都 610065）

摘 要：时间管理算法是决定 RTI 时间管理服务性能的关键。为解决时间管理中常用的 Frederick 算法计算 GALT(greatest available logical time)时可能出现死锁以及仿真系统消息延迟等问题，定义了联邦成员尺度的概念，并结合时间前瞻量的动态调整思想，提出了动态尺度标注算法，并对其进行了分析。分析表明该算法不但减少了消息的延迟时间，还解决了时间管理中的死锁问题。通过在制导弹药飞行视景仿真系统上测试，表明算法改善了仿真效果，提高了仿真系统性能。

关键词：运行支持环境(RTI)；时间管理；时间管理算法；动态尺度标注算法

中图分类号：TP391　　**文献标志码**：A　　**文章编号**：1001-3695(2012)06-2188-04

doi:10.3969/j.issn.1001-3695.2012.06.049

New dynamic scale labeling algorithm in RTI time management

DU Xing-yue[1a]，LU Yu[1b]，CHEN Li-yun[1a]，DANG Ruo-wen[2]

(1. *a. Dept. of Computer Engineering*, *b. Dept. of Training*, *Ordnance Engineering College*, *Shijiazhuang* 050003, *China*; 2. *School of Electrical Engineering & Information*, *Sichuan University*, *Chengdu* 610065, *China*)

Abstract: Time managing algorithm is an importantly sticking point to assure the service of time management. To solve the problems of accidental deadlock while calculating GALT in Frederick algorithm and message delay in the simulation system, this paper defined a new concept of federator scale and used the dynamic lookahead adjusting method, gave out and analyed the dynamic scale labeling algorithm. The analysis result shows that the dynamic scale labeling algorithm not only reduces the message delaying time but also solves the accidental deadlock problem in time management. The algorithm had been tested on the guided projectile navigation scene simulation system. The result indicates that the dynamic scale labeling algorithm improves both the effectiveness and capabilities of the simulation system.

Key words: runtime infrastructure(RTI); time management; time managing algorithm; dynamic scale labeling algorithm

0 引言

HLA 作为一种软件体系结构，为仿真应用的开发提供了构造和描述仿真的通用框架，提高了仿真应用之间的互操作性，促进了模型在不同领域的可重用性[1]。HLA 主要由规则(rules)、对象模型模板(object model template，OMT)以及运行时间支撑系统的接口规范(interface specification)三部分组成。

运行时间支撑(RTI)系统作为 HLA 接口规范规定服务的软件实现，它的主要功能有六种[2]：联邦管理、对象管理、声明管理、时间管理、所有权管理和数据分发管理。其中时间管理(time management，TM)描述的是联邦运行过程中联邦成员仿真时间的推进机制，且该时间推进机制必须与其他联邦成员的数据交换相协调，以确保联邦成员发送与接收消息在时间上的一致性及逻辑上的正确性。时间管理是 RTI 能否快速实时运行的重要影响因素，自 DMSO 公布 HLA 以来，时间管理算法一直是其研究的重点和难点[3]。

Frederick 算法是时间管理的常用算法，该算法在通常情况下能够比较好地工作，但在特殊情况下，该算法会造成死锁。针对死锁问题，文献[4]中提出了身高测量法，文献[5]中提出了动态滑模法。虽然上述两种方法很好地解决了 Frederick 算法中的死锁问题，但是在算法设计时未考虑到算法的效率问题，仿真中消息延迟的问题仍然存在。针对此问题，本文基于 IEEE 1516 技术标准，结合身高测量法以及动态时间管理算法[2]提出了动态尺度标注法，以预防或解决计算 GALT 时死锁问题，并且动态调整时间前瞻量 lookahead，减少消息延迟时间，提高仿真系统性能。

1 基本术语和死锁介绍

1.1 基本术语

定义 1　GALT 是指联邦成员能够推进的最大逻辑时间。

定义 2　minTSO(time stamp order)是指联邦成员消息队列中已有消息的最小时标。

定义 3　输出时间(output time)是指 RTI 计算出的联邦成员能够发送的消息时标的最大下界。

定义 4　挂起是指联邦成员在请求时间推进时因为没有得到 RTI 的允许而处于暂停状态。

收稿日期：2011-11-12；**修回日期**：2011-12-19

作者简介：杜星玥(1988-)，男，四川蓬溪人，硕士研究生，主要研究方向为计算机仿真(406436953@qq.com)；卢昱(1960-)，男，河南洛阳人，教授，博导，博士，主要研究方向为计算机仿真；陈立云(1969-)，男，湖南永州人，副教授，硕士，主要研究方向为计算机仿真；党若雯(1990-)，女，山西太原人，主要研究方向为电动化.

定义 5　死锁是指系统中有相互约束关系的联邦成员均处于挂起状态。

以上定义在文献[4,5]中已经详细证明,这里不再证明。需要强调的是定义 5 中谈到的死锁是由于计算 GALT 引起的。1.2 节将对死锁问题进行详细介绍。

1.2　Frederick 算法的死锁问题

Frederick 算法较好地解决了联邦成员请求时间推进时的 GALT 计算问题,并在绝大多数情况下非常实用和有效,但是该算法却不能避免死锁的产生。下面主要对其存在的死锁问题进行介绍。

例如在一个只有两个成员的联邦执行中,两个成员都是既"时间控制"又"时间受限",并且他们相互预订了对方的"时戳"消息,因此在时间推进时相互约束。

如图 1 所示,两个联邦成员的当前逻辑时间分别为 $T(1)$ 和 $T(2)$,联邦成员 1 请求推进到时间 $T_R(1)$。根据 Frederick 算法(该算法在文献[4,8]中有详细介绍,限于篇幅,此处不再复述)中的 GALT 计算式可知 GALT(1) = OPT(2) = $T(2)$ + lookahead(2),又由 GALT 对"时间受限"成员时间推进请求的限制可知,当 $T_R(1) >$ GALT(1)时,推进请求将不被允许,直到 GALT(1)的值增大到大于 $T_R(1)$ 或 minTSO(1)。因此成员 1 在 NMR 请求挂起状态。随后,联邦成员 2 提出 NMR/NMRA 调用,请求推进到 $T_R(2)$,假设 $T_R(2) > T(2)$ + lookahead(2)和联邦成员 2 中没有小于 $T_R(2)$ 的 TSO 消息。根据 Frederick 算法中的 GALT 计算式可知:GALT(2) = OPT(1) = min{T_R(1) + lookahead(1),minTSO(1),GALT(1)}。可以看出,要计算 GALT(2)的值必须确定 GALT(1)的值。则此时重新计算 GALT(1) = OPT(2),由于此时联邦成员 2 的请求还没被 RTI 准许,也处于挂起状态,则 OPT(2) = min{$T_R(2)$ + lookahead(2),minTSO(2),GALT(2)},即 GALT(1) = min{$T_R(2)$ + lookahead(2),minTSO(2),GALT(2)}。不难看出,GALT(1)的计算依赖于 GALT(2),并且 GALT(2)的计算依赖于 GALT(1),则出现死循环,即死锁状态。

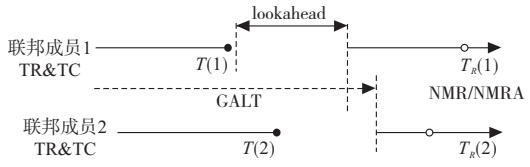

图 1　具有两个联邦成员的死锁示例

针对死锁状态,文献[4]中的身高测量法以及文献[5]的动态滑模法都提出了解锁方法,并都取得了一定解锁效果,但是算法没有解决消息延迟的问题。因此,为了同时解决 GALT 算法死锁和消息延迟问题,本文提出了一种新型的动态尺度标注算法。

2　动态尺度标注算法

2.1　算法的理论依据

规范中指出,仿真系统中各联邦成员的时间前瞻量 lookahead 由应用本身决定,但没有对其进行详细定义。设联邦成员的时间前瞻量 lookahead 值为 L,联邦成员产生事件 e 时的当前逻辑时间为 T,则根据 HLA 协议规定,对于事件 e,其时戳 t_{TS} 应该满足 $t_{TS} \geqslant T + L$,则可得出 $L \leqslant t_{TS} - T$。联邦成员的时间前瞻量 lookahead 值为所有事件时戳值与其当前逻辑时间之差的最小值,则对于任意联邦成员 i,可以总结出:

$$L = \min(t_{TSi} - T_i) \quad e_i \in E$$

其中:E 为联邦成员所有事件的集合;T_i 为产生事件 e_i 的逻辑时间;t_{TSi} 为事件 e_i 的时戳值。

下面给出 HLA 时间管理规范的两条推论[6,7]和一些重要结论[4,5]。

推论 1　如果时间控制联邦成员的 lookahead 设置远小于事件时戳的增加量,则时间受限联邦成员的 RTI 代理在接收到该事件时,需要多次 GALT 的计算才能处理该事件。

推论 2　时间控制联邦成员的 lookahead 设置不能大于事件时戳可能的增量,否则在逻辑上错误,需要事先调整联邦成员的 lookahead 设置。

引理 1　如果所有联邦成员均因为请求时间推进而处于死锁状态,那么对于任意联邦成员 i,GALT 满足公式:

a) $\mathrm{GALT}(i) \leqslant T_R(i)$;

b) $\mathrm{GALT}(i) \leqslant T_R(i) + \mathrm{lookahead}(i)$;

c)如果联邦成员处于 NMR/NMRA 状态,则 $\mathrm{GALT}(i) \leqslant \mathrm{minTSO}(i)$;

d)如果联邦成员处于 NMR/NMRA 状态,则 $\mathrm{GALT}(i) \leqslant \min\{T_R(i) + \mathrm{lookahead}(i), \mathrm{minTSO}(i)\}$。

引理 2　如果所有联邦成员均因为请求时间推进而处于死锁状态,那么对于任意联邦成员 i,输出时间满足公式:

a) 如果联邦成员处于 TAR/TARA 状态,则 $\mathrm{OPT}(i) \geqslant \mathrm{GALT}(i)$;

b)如果联邦成员处于 NMR/NMRA 状态,则 $\mathrm{OPT}(i) = \mathrm{GALT}(i)$;

c)不论联邦成员处于何种挂起状态,都有 $\mathrm{OPT}(i) \geqslant \mathrm{GALT}(i)$,且 $\mathrm{OPT}(i) \leqslant T_R(i) + \mathrm{lookahead}(i)$。

定理 1　如果所有联邦成员均因为请求时间推进而处于死锁状态,那么所有联邦成员的 GALT 相等。

定理 2　如果所有联邦成员均因为请求时间推进而处于死锁状态,那么至少存在两个输出时间最小且相等的联邦成员,它们的输出时间等于其 GALT,且它们是因为 NMR 或 NMRA 请求而被挂起的。

定理 3　采用 Frederick 算法的联邦,联邦成员间既相互控制又相互受限。在联邦运行过程中,如果进行到下面的一种状态,则该联邦运行将处于死锁状态:a)存在至少两个因 NMR 或 NMRA 请求而被挂起的联邦成员,满足 $T_R >$ GALT,且 minTSO > GALT;b)在满足上面要求的联邦成员中,存在两个联邦成员,其输出时间是所有联邦成员中最小的,并且彼此相等;c)上述两个联邦成员的输出时间与它们的 GALT 相等。

以上引理和定理在文献[4,8]中已经详细证明,限于篇幅,这里不再介绍。

2.2　算法介绍

结合 Frederick 算法存在的死锁问题以及身高测量法和动态滑模法中未解决消息延迟的问题,本文提出了一种动态尺度标注算法。这种算法不仅很好地解决了 Frederick 算法死锁的

问题,还很好地减少了消息延迟,使事件传输效率得到了提高。

下面首先给出联邦成员尺度(scale)的定义。

定义 6 尺度可定义为

a)如果联邦成员处于事件准许状态:

$$S(i) = T(i) + \text{lookahead}(i)$$

b)如果联邦成员处于 TAR 或 TARA 状态:

$$S(i) = T_R(i) + \text{lookahead}(i)$$

c)如果联邦成员处于 NMR 或 NMRA 状态:

$$S(i) = \min\{T_R(i) + \text{lookahead}(i), \text{minTSO}(i)\}$$

其中:$S(i)$表示联邦成员的尺度;$T(i)$表示联邦成员当前逻辑时间;lookahead(i)表示联邦成员的时间前瞻量的值;$T_R(i)$表示联邦成员请求推进的时间值;minTSO(i)表示联邦成员 i 消息队列中的最小时戳值。

根据 2.1 节理论知识以及上述定义,动态尺度标注算法步骤可描述如下。

1)时间前瞻量 lookahead 动态调整

算法的基本思想是将联邦成员的 p 个具有不同时间前瞻量和时戳增量的事件集合 E 分成不相交的 q 组,每个事件子集组落入相应的时间 T_i,针对时间段内的事件进行联邦 GALT_i 和 lookahead_i 的计算,逐步推进联邦时间。具体可分步表示为:首先将仿真时间分成 q 段,即 $T = (T_1, T_2, \cdots, T_q)$;然后针对 q 个时间段,将 p 个事件在时间上分为互不相交的 q 组,即 $E = (E_1, E_2, \cdots, E_q)$。在时间段 $T_i(i = 1, 2, \cdots, q)$内有

$$\text{lookahead}_i = \min(t_{\text{TS}j} - T_j) \quad j = 1, 2, \cdots, k$$

其中:k 为时间段 T_i 内的事件总数;$t_{\text{TS}j}$为事件的逻辑时间;T_j 为事件的产生时间。

2)尺度 $S(i)$的计算

在计算 $S(i)$时,使用的均是步骤 1)中计算出的时间前瞻量 lookahead_i 的值,其算法可表示为

```
S(i)
{if(联邦成员 j 处于时间准许状态) then
   {return(T(j) + lookahead(j));}
   else if(联邦成员 j 处于 TAR 或 TARA 状态) then
   {return(T_R(j) + lookahead(j));}
   else if(联邦成员 j 处于 NMR 或 NMRA 状态) then
/*设所有对成员 j 时间推进有直接约束的成员为 j1,j2,…,jn */
/*令 m = min{T_R(j) + lookahead(j), minTSO(j)} */
     {for(k = 1;k <= n;k++)
       {m = min(m,s_jk)}
     return(m);}
}
```

3)GALT 计算

对于任意联邦成员 i,其 GALT 取决于其他联邦成员的尺度,公式为

$$\text{GALT}(i) = \min\{S(j)\} \quad i \neq j$$

动态尺度标注算法中规定:如果一个联邦成员请求时间推进时的尺度最小,则该联邦成员的请求总能先得到满足。

2.3 算法分析

2.3.1 算法对消息延迟问题的解决

动态尺度标注算法使事件集分段,然后再落入相应的时间分段。如算法描述的将事件集 E 分成了不相交的 q 组,假设每个子集的事件数为 $N_i(i \in [1, q])$,则每个事件集计算 GALT 的次数分别为$(1 + N_i) \times N_i/2, i \in [1, q]$。假设计算 GALT 的总次数为 N,则有

$$N = \sum_{1}^{q} \frac{N_i}{2} \times (N_i - 1) \quad i \in [1, q]$$

其中:N 是使用了动态尺度标注算法得出的 GALT 计算总次数。如果不使用动态尺度标注算法,对于同一事件集 E,其事件总数为 K,且满足 $K = N_1 + N_2 + \cdots + N_q$。假设 GALT 计算总次数为 N',则有 $N' = (K-1)^2$,即 $N' = (N_1 + N_2 + \cdots + N_q - 1)^2$。不难看出,$N' >> N$,即使用动态尺度标注算法后,大大减少了对 GALT 的计算次数。联邦成员同步时间开销公式为

$$T_w = N \times \gamma$$

其中:T_w 为联邦成员同步时间开销;N 为计算 GALT 的总次数;γ 为一个联邦成员节点从发起 GALT 计算到取得其他节点状态并计算出 GALT 所花的时间。当 γ 一定时,计算 GALT 的总次数 N 越少,则联邦成员同步时间开销 T_w 越少。动态尺度标注算法运用了时间前瞻量 lookahead 动态调整,使 GALT 的计算次数 N 较之前的计算次数 N'大大减少。又由于在同一仿真系统中的拓扑结构是不变的,即 γ 不变,从而使用动态尺度标注算法降低了联邦成员的时间同步开销,减少了消息的延迟时间,改善了仿真系统效果,提高了仿真系统性能。

2.3.2 算法对死锁问题的解决

使用动态尺度标注算法不会导致死锁。下面结合 1.2 节中的例子加以说明。根据算法第二步分别计算联邦成员 1 和 2 的尺度为

$$S(1) = \min\{T_R(1) + \text{lookahead}(1), \text{minTSO}(1)\}$$
$$S(2) = \min\{T_R(2) + \text{lookahead}(2), \text{minTSO}(2)\}$$

根据动态尺度标注算法第三步中的规定,如果一个联邦成员请求时间推进时的尺度最小,则该联邦成员的请求总能先得到满足。为讨论使用动态尺度标注算法对死锁的解决问题,首先假设 $S(1) > S(2)$,则可计算出联邦成员 2 的 GALT 为 $\text{GALT}(2) = \min\{S(1)\} = \min\{T_R(1) + \text{lookahead}(1), \text{minTSO}(1)\}$;如果 $\text{minTSO}(1) > T_R(1) + \text{lookahead}(1)$,则有 $\text{GALT}(2) = T_R(1) + \text{lookahead}(1)$,可得 $\text{GALT}(2) > T_R(2)$,则联邦成员 2 可顺利移动到 $T_R(2)$时刻,死锁被打破。如果 $\text{minTSO}(1) < T_R(1) + \text{lookahead}(1)$,则有 $\text{GALT}(2) = \text{minTSO}(1)$,此时又可分为两种情况:若 $\text{minTSO}(1) > T_R(2)$,则联邦成员 2 可顺利移动到 $T_R(2)$时刻,死锁被打破;若 $\text{minTSO}(1) < T_R(2)$,则有 $\text{minTSO}(1) > \text{minTSO}(2)$,进而得出 $\text{GALT}(2) > \text{minTSO}(2)$,则联邦成员 2 可顺利移动到 minTSO(2)时刻,死锁被打破。$S(1) < S(2)$时算法对死锁的解决问题分析方法与 $S(1) > S(2)$类似,这里不再复述。

死锁产生的根本原因是在计算输出时间时增加了对 GALT 的比较。动态尺度标注算法定义了尺度,用尺度代替了输出时间,不再增加 GALT 的比较,在根本上避免和解决了死锁的发生。

2.3.3 算法性能测试

本文在制导弹药飞行视景仿真系统中对动态尺度标注算法进行了测试。制导弹药飞行视景仿真系统由五个联邦成员构成,本文设定实验中,系统使用相同的联邦成员和仿真步长,分别采用动态尺度标注算法以及不采用该算法,总共进行了六

次对比性实验,完成制导弹药飞行视景仿真过程。实验过程中按照处理事件由少逐渐增多,并且每次系统处理事件数目一致的实验要求,主要对时间消耗以及是否出现死锁两个问题进行了对比分析研究,得到的实验结果如表 1 所示。

表 1　动态尺度标注算法对仿真系统的影响

仿真处理事件数		96	203	500	910	1 794	3 280
未采用动态尺度标注算法	时间消耗/s	90.3	182.7	435.0	773.5	1 524.9	2 755.2
	是否出现死锁	否	否	是	否	否	是
采用动态尺度标注算法	时间消耗/s	50.8	104.6	218.1	398.0	844.3	1 801.9
	是否出现死锁	否	否	否	否	否	否

从表 1 的数据可以看出,未采用动态尺度标注算法在第三次和第六次实验时,系统出现了死锁,即系统停止弹道解算和数据传输,制导弹药停在三维显示成员屏幕上。表 1 中对应这两次的时间消耗数据 435 s 和2 755.2 s是假设实验完成的理论估计值。对比观察,系统采用动态尺度标注算法进行的六次实验没有出现死锁,即解决了 Frederick 算法中的死锁问题,还明显减少了处理所有事件的时间消耗,改良了系统性能。对比使用动态尺度标注算法前后所花费的时间与处理的事件总数,还可以得出另一条重要结论:随着事件数增加,处理事件所花费的时间在减少。处理事件总数与时间消耗关系如图 2 所示。

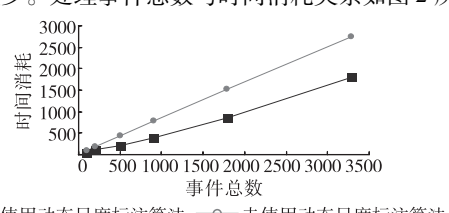

图 2　处理事件总数与时间消耗关系

3　结束语

本文主要对 HLA/RTI 时间管理算法问题进行了研究,针对时间管理中常用的 Frederick 算法存在死锁以及仿真中消息延迟等问题,提出了动态尺度标注算法,其本质上仍是对 Frederick 算法的改进。最重要的改进之处是采用分段动态计算时间前瞻量 lookahead 和 GALT 的值,并定义了尺度这一概念以取代 Frederick 算法中的输出时间。动态尺度标注算法不仅减少了 GALT 的计算次数,还避免了死锁的发生。在制导弹药飞行视景仿真系统中也取得了不错的测试成果。

参考文献:

[1] 张家祥. HLA 仿真技术应用教程[M]. 北京:国防工业出版社,2007.

[2] 窦志武. 基于高层体系结构分布交互仿真的应用方法研究[M]. 北京:经济科学出版社,2009.

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

[4] 刘步权,王怀民,姚益平. 一种无死锁的时间管理算法[J]. 软件学报,2003,14(9):1515-1522.

[5] 周晓滨,方洋旺,许勇,等. 基于动态滑模的无死锁时间管理算法[J]. 计算机工程,2008,34(5):14-19.

[6] FUJIMOTO R M. Parallel simulation:parallel and distributed simulations systems[C]//Proc of the 33rd Conference on Winter Simulation. Washington DC:IEEE Computer Society,2001:147-157.

[7] CAROTHERS C D,FUJIMOTO R Weatherly R M,*et al*. Design and implementation of HLA time management in the RTI version F. 0 [C]//Proc of the 29th Conference on Winter Simulation. Washington DC:IEEE Computer Society,1997:373-380.

[8] 唐京桥,侯朝桢. HLA 中时间管理算法死锁的规律性[J]. 计算机工程,2005,31(15):27-29.

[9] 龚爱珍,艾丽蓉,王琼. 基于同步和异步时间管理的混合时间管理算法[J]. 计算机技术与发展,2011,21(7):32-35.

[10] 曹海旺,薛朝改,黄建国. HLA 步进联邦时间管理性能的影响因素分析[J]. 计算机应用研究,2011,28(2):521-523.