第 38 卷 第 4 期
Vol.38 No.4

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

·开发研究与设计技术· 文章编号：1000—3428(2012)04—0251—03 文献标识码：A 中图分类号：TP312

一种基于广义表的 HLA 乐观时间推进算法

韩晓东，翟正军，丁 超

(西北工业大学计算机学院，西安 710072)

摘 要：针对保守时间推进机制缺乏灵活性、仿真时间长等问题，研究高层体系结构中的时间管理服务，通过分析和比较，证明乐观时间推进算法可以弥补保守时间推进机制的不足。引入广义表存储结构，实现乐观时间推进机制中的存储和回退等关键技术。测试结果表明，该算法可以减少系统的仿真时间，提高系统的实时性。

关键词：高层体系结构；保守时间推进；乐观时间推进；广义表；回退

HLA Optimistic Time Advance Arithmetic Based on Generalized List

HAN Xiao-dong, ZHAI Zheng-jun, DING Chao

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

【Abstract】This paper makes a research on time management services of High Level Architecture(HLA). It proves optimistic time advance can solve lack of agility and a long emluator time of conservative time advance via analyzing and comparing of conservative time advance and optimistic time advance; And it introduces a generalized list storage structure. Its two-dimensional storage effectively resolves key storage and retraction technique of optimistic time advance. By testing and verification of the arithmetic, it proves that the arithmetic shortens the emluator time and heightens real-time of the simulation system.

【Key words】High Level Architecture(HLA); conservative time advance; optimistic time advance; generalized list; retract

DOI: 10.3969/j.issn.1000-3428.2012.04.082

1 概述

高层体系结构(High Level Architecture, HLA)中的时间管理服务能够保证仿真世界事件发生的顺序与现实世界事件发生的顺序保持一致。时间管理服务是 HLA 能够仿真出一个真实、有意义系统的关键。目前一般采用的时间推进机制是保守时间推进机制，在实时性要求很高的仿真系统中，保守时间推进机制有时会影响整个联邦执行的实时性，乐观时间推进机制的提出可以提高仿真系统的实时性。而目前 HLA 领域对乐观时间推进机制的研究只是提出了上层的框架，对整个机制算法的实现相对较少，更缺乏不同算法实现性能优劣的判定[1]。为提高仿真系统的实时性，本文给出了基于广义表的实现算法。

2 保守时间推进与乐观时间推进比较

保守联邦成员是指采用保守时间推进机制[2]的联邦成员；乐观联邦成员是指采用乐观时间推进机制的联邦成员。

保守时间推进保证联邦成员只会接收到时间戳顺序(Time Stamp Order, TSO)越来越大的事件，运行支撑环境(Run Time Infrastructure, RTI)不会向保守联邦成员发送过去的 TSO 事件。TSO 队列内部肯定是按照 TSO 顺序排列的，TSO 小的在队首，TSO 大的在队尾。TSO 队列外面的 TSO 事件有可能不是按 TSO 顺序排列的，但是它们肯定都大于队列中的所有 TSO 事件，而且会按照 TSO 顺序入队。

在保守时间推进机制中，时间受限成员只有在请求推进过程中才能接收到当前仿真时间和请求推进的时间加上前瞻量之间的 TSO 消息，而时间受限成员的推进受限于时间调节成员。因此，只有时间调节成员先进行推进，然后时间受限成员才能够在推进过程中接收到 TSO 消息。保守时间推进机制具体推进原理如图 1 所示。

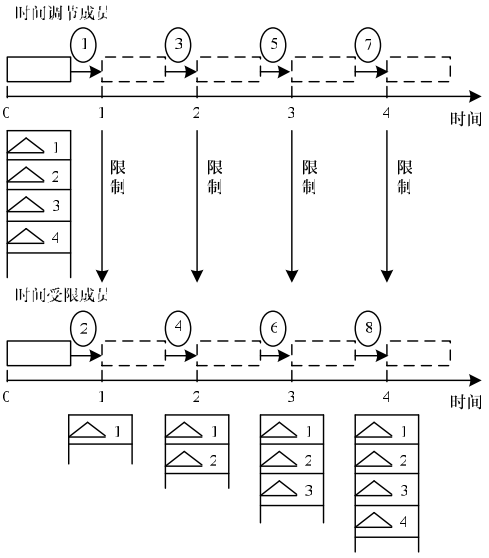

图 1 保守时间推进机制原理

在图 1 中，“Δ1”代表时间戳为 1 的 TSO 消息，其他类推，时间调节成员在当前时刻(TSO=0)只能发送大于当前仿真时间的 TSO 的消息，时间受限成员只有在推进过程中才能接收到 TSO 消息，而时间调节成员又限制了时间受限成员的推进，因此，需要时间调节成员先进行推进。

乐观联邦成员总是希望尽早得到 TSO 事件，RTI 可以满足乐观联邦成员的这一要求。乐观联邦成员可以接收超前的

作者简介：韩晓东(1986－)，男，硕士，主研方向：计算机测控仿真，嵌入式系统；翟正军，教授；丁 超，本科生

收稿日期：2011-07-15 **E-mail：**hxiaodong623@163.com

TSO 事件，且 TSO 队列内部肯定是按照 TSO 顺序排列的，TSO 小的在队首，TSO 大的在队尾。TSO 队列外面的 TSO 事件有可能不是按 TSO 顺序排列的，而且有可能小于队列中的 TSO 事件，会按照 TSO 顺序由小到大进入 TSO 队列并且排在队尾。

乐观时间推进机制中的时间受限成员和保守推进机制中的时间受限成员相同点在于只有在推进过程中才能接收到 TSO 消息，不同点是乐观时间推进机制下时间受限成员不受时间调节成员的限制，可以超前推进，但是能够成功推进的仿真时间是请求推进的仿真时间和接收到的 TSO 消息中时间戳最小点。保守时间推进机制具体推进原理如图 2 所示。

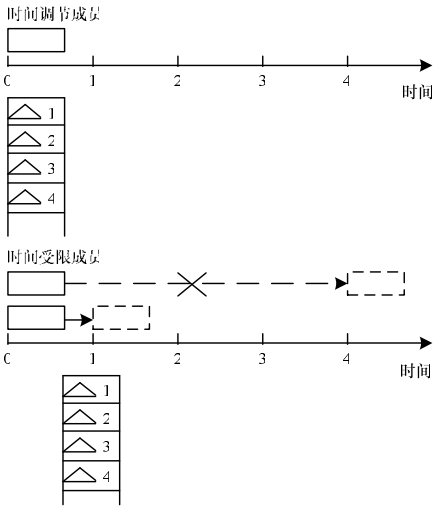

图 2 乐观时间推进机制原理

在图 2 中，时间受限成员请求推进到仿真时间戳 4 处，然而因为 TSO 队列中的最小时间戳为 1，所以时间受限成员最多可以超前推进到时间戳为 1 处。

在某直升机任务效能评估系统[3]中经过测试发现，在对仿真系统实时性要求很高的情况下，若一次数据交互需要经过大量的模型解算，则需要大量模型解算的联邦成员，导致其计算延时很长，结果影响到整个仿真系统的实时性；并且由于解算的复杂性，因此其每次计算延时不同，从而导致整个仿真系统产生抖动。而且在一个复杂的联邦执行中，某一个主控联邦成员(时间调节成员)在对一部分时间受限成员进行时间引导的过程中同时限制了其他时间受限成员，而这部分时间受限成员和主控联邦成员并无事实上的因果关系；还有一部分事件之间并不存在先后因果关系，它们之间互为因果同时发生，这样，保守时间推进机制严格地按照时间戳进行排队反而影响了仿真系统的实际意义。在这种情况下使用乐观时间推进机制可以解决延时、抖动和弱因果关系等问题。

3 乐观时间推进算法流程

超前调度时间是指乐观联邦成员在处理一个大于它当前仿真时间的 TSO 事件所具有的时间戳。由前瞻量[4]和最小时间戳下限的定义可以得到以下 2 个规则：

(1)超前调度时间一定大于等于它的 LBTS(Lower Bound on Time Stamp)。

(2)只有超前调度时间有可能需要回退，其他仿真时间都不能回退。

在图 2 中，当乐观联邦成员处理完 $T1$=3、$T2$=4 事件之后，若发现接收的下一个事件的超前调度时间小于刚处理的超前调度时间，则联邦成员需要回退。乐观时间推进机制的整个流程如图 3 所示。

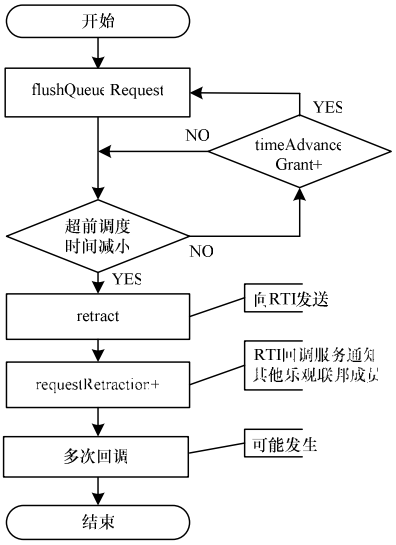

图 3 乐观成员推进算法流程

图 3 中有 2 个问题 HLA 并没有进行详细的设计：

(1)如何准确定位何时向 RTI 发送回退消息。

(2)如何实现乐观联邦成员的事件回退。

多个联邦成员之间的回退在 RTI 服务的协助下完成，发送过去的时间戳的联邦成员负责主动提出回退，RTI 协助将回退命令发送给接收该联邦成员时间戳消息的其他联邦成员；若其他联邦成员又将该时间戳消息发送给了另外的联邦成员，则重复上面的步骤。回退过程的示意图如图 4 所示。图 4 的黑框表示主动要回退的联邦成员；其他所有白框表示被动收到需要回退的联邦成员。

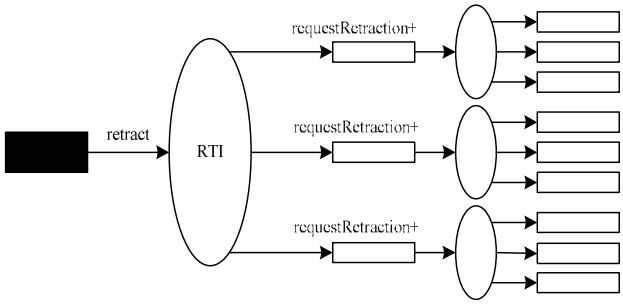

图 4 乐观成员回退过程

4 基于广义表的乐观时间算法实现

存储乐观联邦成员当前的状态为下一步实现回退机制提供依据。需要存储的信息包括从 TSO 队列中接收的超前调度时间和对应的发送该 TSO 消息的联邦成员的句柄及其当前仿真时间。数组索引的优点在于能够迅速访问任何位置的元素，链表的优点在于能够迅速删除任何位置的元素，两者在其他方面都有明显的缺陷。

二叉搜索树虽然在检索和删除方面都很方便，但是它是依赖于关键字的大小来建树的。采用乐观时间推进机制的联邦成员可以作为搜索关键字的是超前调度时间。而超前调度时间在引发回退之前是递增的，一旦接收到小于 TSO 队列中某个超前调度时间，则该超前调度时间之后的事件都应该回退，即对应的信息都应该删除，这样即使建立一棵二叉搜索树，其平衡度也很差，已经退变为一个单向链表。

从上述分析可知，采用单向链表在数据结构复杂度、检

索、删除方面具有平均情况下最好的性能。由于每个联邦成员接收到的消息有可能来自多个联邦成员，因此要为每个联邦成员设计一个二维的单向链表。广义表[5]是一种递归的表，允许表中有表，可以实现二维表，广义表存储表示如图 5 所示。

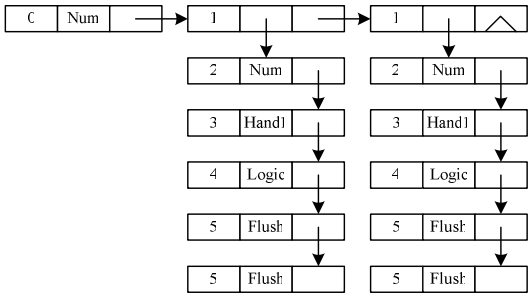

图 5　广义表带头节点的存储表示

在图 5 中，广义表的每一个节点由 3 个域组成：标志域，值域和尾指针域。标志域标识该节点是什么类型的节点：为 0 表示头节点；为 1 表示子表节点；为 2 表示子表表头节点；为 3 表示句柄节点；为 4 表示当前仿真时间节点；为 5 表示 TSO 队列中的超前调度时间节点。值域根据标志域决定为何值。当标志域为 0 或 1 时，尾指针域指向同一层下一个节点的地址，当标志域为其他值时，尾指针域指向子表表头节点的地址。广义表对应的数据结构如下：

```
typedef enum
{
HEAD=0,
LIST=1,
SUBHEAD=2,
SUBHANDLE=3,
SUBLOGICAL=4,
SUBFLUSH=5
} ElemTag ;
class GenListNode
{
private:
  ElemTag utype;
  GenListNode *tlink;
  union
  {
        int head;
        int subHead;
        int subHandle;
        float subLogical;
        float subFlush;
        GenListNode *hlink;
  }value;
};
class GenList
{
private:
  GenListNode *first;
public:
  GenListNode &Head();
  GenListNode *First();
  GenListNode *Next(GenListNode *elem);
  void setNext(GenListNode *elem1,GenListNode *elem2);
  int Remove(int handle,GenListNode *elem);
  int Add(int handle,float logical,GenListNode *elem);
  int Find(int handle,GenListNode *elem);
};
GenList *List;
void retract(MessageRetractionHandle const &  theHandle)
{while (temp->tlink!=NULL)
{
遍历广义表;
if(temp2->value.subHandle== theHandle)
{
  查找与句柄匹配的节点;
  删除该子表之后的所有节点;
}}}
void requestRetraction(MessageRetractionHandle const & theHandle)
{while (temp->tlink!=NULL)
{
  遍历广义表;
if(temp2->value.subHandle== theHandle)
{
  查找与句柄匹配的节点;
  删除该子表之后的所有节点;
}}}
```

当联邦成员接收到小于 TSO 队列中某个超前调度时间的事件时，该联邦成员需要调用 RTI 提供的接口来回退事件，该联邦成员将删除掉在乐观情况下已经计算的超前调度时间大于等于该超前调度时间对应的所有节点。在 retract 和 requestRetraction 中调用删除节点算法，删除节点采用下面的方法实现：

当该乐观联邦成员向 RTI 发送回退事件时，联邦中其他乐观联邦成员将接收到 RTI 发送的回退通知，其他乐观联邦成员也会像引起回退的乐观联邦成员一样进行回退，删除相应的节点。

5　测试结果与结论

测试方法为建立一个有 2 个联邦成员的联邦执行，成员 1 为时间主控成员，成员 2 为时间受限成员。成员 1 模拟 10 个时间调节成员向成员 2 发送 TSO 消息，分别传递不同数量的 TSO 消息，测试在保守时间推进机制、乐观时间推进机制及在回退情况下需要的物理时间。

保守时间推进与乐观时间推进算法耗时比较见表 1。

表 1　保守时间推进与乐观时间推进算法耗时比较

TSO 消息个数	算法耗时/s	
	保守时间推进算法	乐观时间推进算法
20	0.151 724	0.026 793
50	0.360 775	0.065 012
100	0.550 030	0.139 306
…	…	…

由以上测试结果可知，乐观时间推进算法需要的物理时间远少于保守时间推进算法所需的时间，并且在考虑到有可能发生多次回退所需要的时间之后，上述关系仍然成立。

6　结束语

本文对比研究了保守时间推进机制和乐观时间推进机制，发现保守时间推进机制可能会影响整个仿真系统的实时性，在研究 HLA 提供的乐观时间推进机制与保守时间机制协同运行的基础上，设计实现了乐观时间推进机制算法中的存储和回退机制，在功能上使得仿真系统中可以同时存在保守时间推进机制和乐观时间推进机制，在需要大量模型解算的系统中可以增强仿真系统的实时性。　(下转第 256 页)