文章编号:1001 - 2486(2005)03 - 0105 - 05

基于引擎成员的仿真调度问题*

凌云翔[1],邱涤珊[1],张小雷[2],古西睿[1]
(1. 国防科技大学 信息系统与管理学院,湖南 长沙 410073; 2. 中国科技大学 计算机系,安徽 合肥 230027)

摘 要:仿真引擎作为仿真系统的核心,是 HLA 仿真联邦中的特殊联邦成员。论述了基于实体、成员和引擎的层次式仿真调度概念结构,及其采用的基于时间序和事件序的仿真调度方案,将作战实体的时间推进和交战处理都归入仿真引擎,确保了全局时间统一性、实体同步性和交战事件的有序性,而且各种参量的设置灵活,便于控制。介绍了仿真引擎成员的设计与实现。该引擎成员在某指挥自动化作战效能仿真与评估系统中得以应用。

关键词:仿真引擎;成员;调度;层次结构
中图分类号:TP315 文献标识码:A

The Simulation Scheduling Based on the Engine Federate

LING Yun-xiang[1], QIU Di-shan[1], ZHANG Xiao-lei[2], GU Xi-rui[1]
(1. College of Information System and Management, National Univ. of Defense Technology, Changsha 410073, China;
2. College of Computer, University of Science & Technology of China, Hefei 230027, China)

Abstract: The simulation engine (SE) is the most important federate among the federations based on HLA. This paper puts forward a simulation scheduling architecture including SE, federate vessel and entity. And then it raises a hiberarchy simulation schedule arithmetic based on time sequence and event sequence. At last, we introduce the design and implementation of the federate framework of SE, which is applied in efficiency simulation and the evaluation system of C^3I.

Key words: simulation engine; federate; scheduling; hiberarchy architecture

仿真引擎是仿真系统中负责时间推进、调度运行、仿真控制、数据存储,并为仿真运行过程中的态势显示提供交互的仿真运控系统。在研制的 SAESim 仿真系统[3]中,把仿真引擎看作是一个特殊的联邦成员(以下简称成员)。作为仿真环境的核心,它使用基于时间序和事件序的层次式仿真调度算法控制整个系统的运行,控制时间推进,协调功能成员,调用实体模型、规则模型和交战模型,负责剧本的判读,进行交战判断,支持剧本的动态编辑,对仿真数据进行选择存储。联邦中各成员都要受到仿真引擎的控制,根据剧本内容来运行。

1 仿真系统调度结构

在 HLA 仿真系统中,联邦成员的运行可视为一系列计算,联邦相对于 RTI 是一组相互交换带时戳事件的成员集合[1],引擎则是联邦中控制其它成员交换事件的特殊成员,而 RTI 是事件的交换机。仿真引擎就是‘操作系统’;仿真中的各个实体对象就是‘用户’,它向仿真引擎发送时间推进请求或调用交战请求;‘中间件’则是作为实体模型‘容器’(container)的成员,姑且称之为成员容器。成员容器与仿真引擎都是联邦执行中的成员,所以成员容器完全可以胜任仿真引擎的输入输出信息传送,而对于实体对象而言,由于实体对象是与 RTI 无关的功能模块,其要与仿真引擎进行信息交互,就必须通过 RTI 通信接口,成员容器作为‘中间件’,为它提供了这样一个接口,使得实体对象和仿真引擎可以在 RTI 联邦环境下进行信息交互和仿真执行。仿真调度的层次概念结构如图 1 所示。

* 收稿日期:2004 - 12 - 29
基金项目:国家部委资助项目
作者简介:凌云翔(1972—),男,副教授,博士。

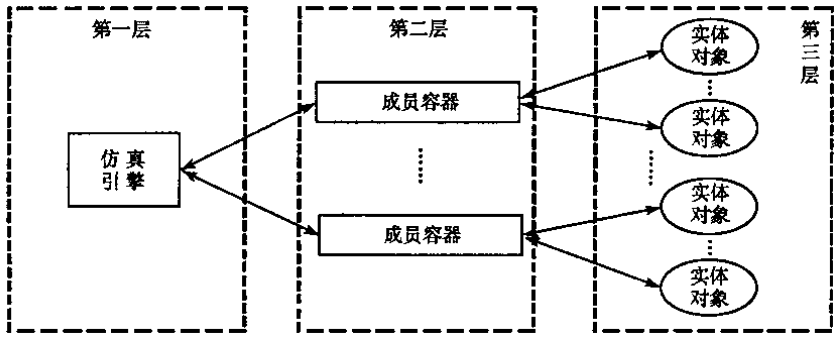

图1 仿真调度层次概念结构图

Fig.1 Hiberarchy simulation schedule architecture

2 仿真调度问题描述

基于上述分布式结构,每个实体对象都是通过局部范围某个节点上运行的成员容器参与到同步仿真工作中去的。从最初的仿真初始化设置,直至最后的效能评估,在整个过程中,系统经历了各种状态的更迭。在实体对象的一次操作下,各仿真模型要进行复杂的交互,才能使整个系统从一种状态转到另一种状态,也就是说状态 $S_i \rightarrow S_{i+1}$ 要经过一个同步的过程,这种过程反映了仿真引擎处理问题的规则,是系统设计和构造时所关心的主要问题。为此,采用层次调度机制,分别在成员容器和仿真引擎两个层次上实施仿真任务的调度。为了便于描述,根据请求性质的分类,把引发时间推进服务的实体对象请求叫做 Time,相应的排队机构叫 Time 队列 把引发事件处理的实体对象请求叫做 Event,相应的排队机构叫 Event 队列,如图2所示。

第一级为仿真引擎的总控管理,包括服务器、时间和事件缓冲队列及其全局调度器。第二级实施成员容器的任务组级管理,每个成员容器 fv_i 包括一个时间推进请求缓冲队列 tq_i 和一个事件请求缓冲队列 eq_i 及其相应调度器,队列的最大空间为 B_i,每个实体对象只能在队列中占据一个位置,故 B_i 为组内可容纳的最大实体对象数目。这里的 B_i 需要大于该成员容器中的实体数目,以防止实体请求丢失。第三级是实体对象,它是被调用的对象。

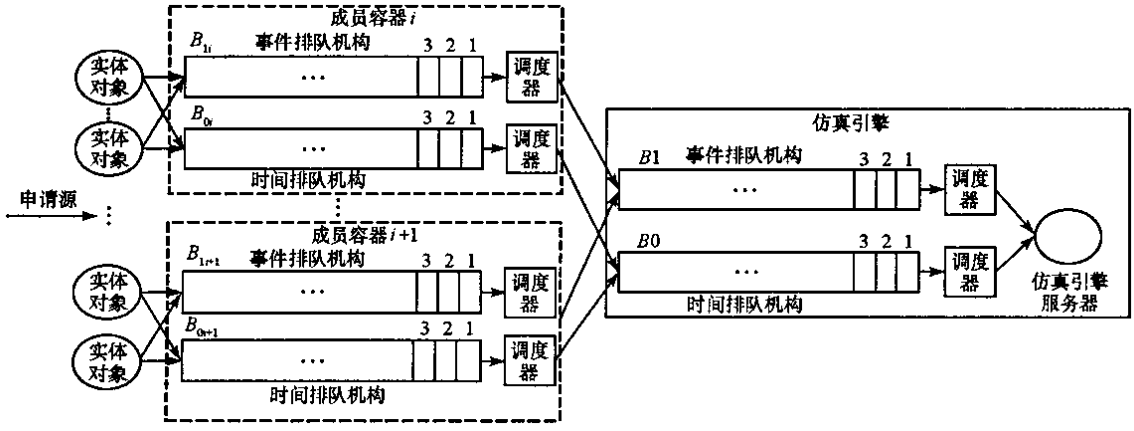

图2 仿真调度机制

Fig.2 Simulation schedule mechanism

为了体现仿真中各层次对象的动态变化过程,引入仿真步(simulation step)的概念[2],即把实体、成员和引擎的运作看作类似于流水操作的一系列仿真步组成。如图3所示。

实体 i 的每帧可以分为四段,$t_{i1} \sim t_{i4}$ 为各仿真步所需时间:

(1) a_{i1} 表示实体 i 方的信息接收,包括可能的时间推进量命令或返回的交战结果信息;

(2) a_{i2} 表示本地处理,如时钟推进,根据交战结果修改相应属性等;

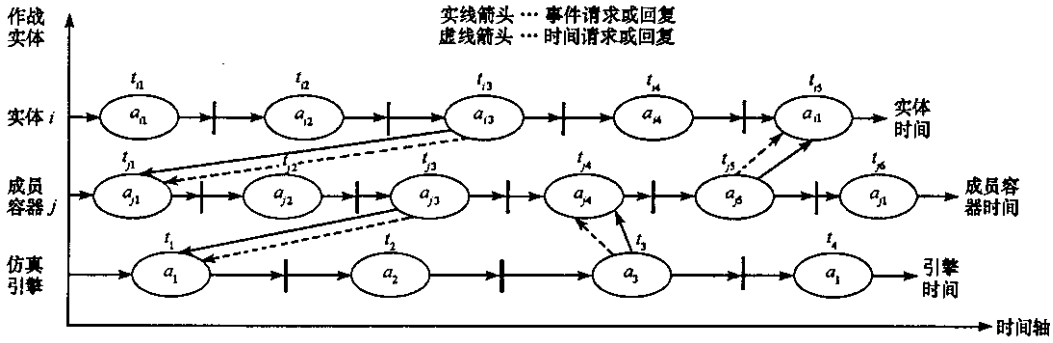

图 3 基于交战行为的实体同步

Fig.3 Entity synchronization based on warfare interaction

(3) a_{i3} 表示发送请求,根据最新的实体状态变化向上一层发送时间推进或调用交战模型的请求;

(4) a_{i4} 等待上层的信息处理。

成员容器 j 的每帧可以分为五段,$t_{j1} \sim t_{j5}$为各仿真步所需时间:

(1) a_{j1} 表示接收实体对象传送过来的请求;

(2) a_{j2} 表示对接收到的实体请求进行初步处理,将时间推进请求和事件处理请求分开并排入不同的缓冲队列;

(3) a_{j3} 表示从两个缓冲队列中分别向引擎发送实体对象的未处理请求信息,然后等待;

(4) a_{j4} 表示接收仿真引擎的处理后信息;

(5) a_{j5} 表示向实体对象发送仿真引擎发送过来的处理信息。

仿真引擎每帧分为三段,$t_1 \sim t_3$ 为各仿真步所需时间:

(1) a_1 表示接收成员容器关于实体对象的请求信息;

(2) a_2 表示对这些信息进行分类处理,分别放入时间缓冲队列和事件缓冲队列,按调度规则进行规则判断、交战模型调用和剧本判读;

(3) a_3 表示向成员容器发送处理后的结果信息。

3 基于时间序和事件序的层次式仿真调度算法描述

为了便于叙述,记 TLe_i 表示实体对象 e_i 的实体时间,$TLfv_j$ 表示成员容器 fv_i 的成员时间,$TLse_j$ 表示仿真引擎的仿真时间。

(1) 在仿真实体 e_i 方

① a_{i1}阶段,e_i 接收由 fv_j 传输来的同步结果数据,如果为时间数据,则传输的为时间推进量 GrantTime 如果是交战结果信息,则传输的是该实体对象交战请求的交战结果;

② a_{i2}阶段,对接收的同步结果数据进行判断,对于交战结果信息,实体对象修改模型的对应属性,以适应交战后的状态;

③ 对于时间推进信息,则修改 $TLe_i = TLe_i + GrantTime$;

④ 经 a_{i2}阶段之后,组织发送该帧的请求数据包(时戳为 TLe_i)进入临时保留场所;

⑤ 仿真实体 e_i 等待。

(2) 在成员 fv_j 方

① aj_1 阶段开始,fv_j 在临时保留场所接收由 e_i 传输来的同步请求数据(时戳为 TLe_i);

② e_i 请求在临时保留场所接受 fv_j 的判断,如果 e_i 发出的是时间推进请求,则转发给队列 tq_j,否则进入队列 eq_j 等待 fv_j 调度;

③ 在 a_{j3}阶段,fv_j 的时间队列调度器和事件队列调度器分别按先进先出的原则将临时保留场所排列的请求包(时戳为 TLe_i)调出,然后发送给临时总保留场所 tq 或 eq,记时戳为 TLe_i ;成员容器 fv_j 等待;

④ a_{j4}阶段,接收仿真引擎的回传结果数据包;

⑤ 将结果数据包回传给实体对象 e_i。

(3) 在引擎 *se* 方

① a_1 阶段开始,*se* 在临时总保留场所接收由 fvj 传输来的同步请求数据(时戳为 TLe_i),将从各分时间队列中传来的时间请求数据放入总时间队列 tq,按请求时间推进的值由近到远排序;将从各分事件队列中的交战请求数据放入队列 eq,按时戳大小排序,等待 *se* 调度;

② a_2 阶段,*se* 选择时间队列中拥有最小时间的成员 fv_j,请求作为服务对象,准备时间推进;

③ *se* 扫描事件队列,选择在当前时间和将要推进到的时间点之间具有最小时间点的事件请求作为服务对象,进行规则模型判断,调用交战模型。从事件队列中删除该请求,该事件请求可能已经接收到与此次交战相关的实体对象集合 A 中所有实体对象传输来的同步数据,也可能仍有未到者,此时仿真引擎和该请求仍需等待,直至全部相关数据到达才开始服务,若等至最大容忍时间 α_j 时仍未到齐,则发送同步信息 *NEGfv*;

④ *se* 反复扫描事件队列,直至在当前时间和将要推进到的时间点之间的事件请求全部被执行完,*se* 事件请求服务完毕;

⑤ 时间推进,修改时钟,记时间推进量为 *GrantTime*;

⑥ a_3 阶段,*se* 把同步结果数据(交战结果数据和时间推进量)回传给各相关成员容器。

4 仿真引擎成员的设计实现

仿真引擎是一个特殊的成员,由仿真调度器、仿真控制器和数据记录器组成,采用基于时间序和事件序的层次式调度方案,分别实现作战方案映射、仿真调度控制、模型的加入退出、剧本解析执行、仿真数据存储等功能。它的输入信息为作战方案,输出为各种有效的中间数据和结果数据。它在仿真过程中与 *RTI* 进行交互,为用户屏蔽了 *RTI* 的接口。其具体结构如图 4 所示。

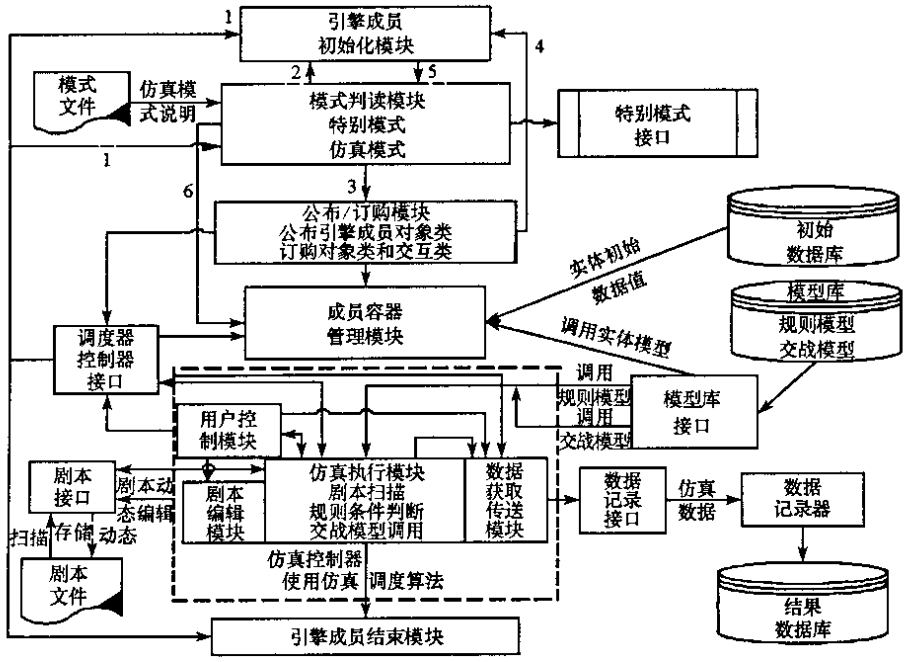

图 4 仿真引擎成员实现结构

Fig.4 Implement architecture of simulation engine federate

调度器是仿真引擎的调度中心,包含初始化模块、模式判读模块、公布/订购模块、成员容器管理模块和引擎成员结束模块,主要完成引擎成员启动/销毁、作战方案解读、信息公布订购、各应用成员及实体模型的管理等。仿真控制器是仿真引擎的控制中枢,仿真过程中的所有动作都是在仿真控制器的控

制之下进行的,包括用户控制、仿真执行、剧本动态编辑以及数据获取。而数据记录器主要用于仿真评估数据的分类存储。它从仿真控制器中获取需要存储的数据类型说明,创建存储记录表,并在仿真过程中通过跟仿真控制器的交互不断得到记录数据值,然后采用集中方式将其分类存储到结果数据库中。

5 结束语

把仿真引擎看作 HLA 仿真联邦中的特殊成员,所采用的基于时间序和事件序的层次式仿真方案将作战实体的时间推进和交战处理都归入仿真引擎,确保了全局时间统一性、实体同步性和交战事件的有序性,而且各种参量的设置灵活,便于控制。随着仿真规模的扩大、作战实体的增加,解决调度效率问题的有效方法是对层次式调度机构进行分级处理,减轻引擎压力。有关引擎的可扩展性问题在此不做赘述。

参 考 文 献:

[1] 柏彦奇. 联邦式作战仿真[M]. 北京:国防大学出版社,2001.
[2] 凌云翔,等.基于 HLA-RTI 的协同仿真模型[J]. 计算机研究与发展,1999,(36):267 – 272.
[3] 祝江汉,凌云翔,邱涤珊. 指挥自动化系统效能仿真环境研究[J]. 计算机科学与工程, 2004,(1).
[4] 周彦,戴剑伟,等.HLA 仿真程序设计[M].北京:电子工业出版社,2002.
[5] Defense Modeling and Simulation Office. HLA Federation Design and Development and Execution Process(FEDEP) Model(Version 1.2)[R]., 1998.

(上接第 48 页)

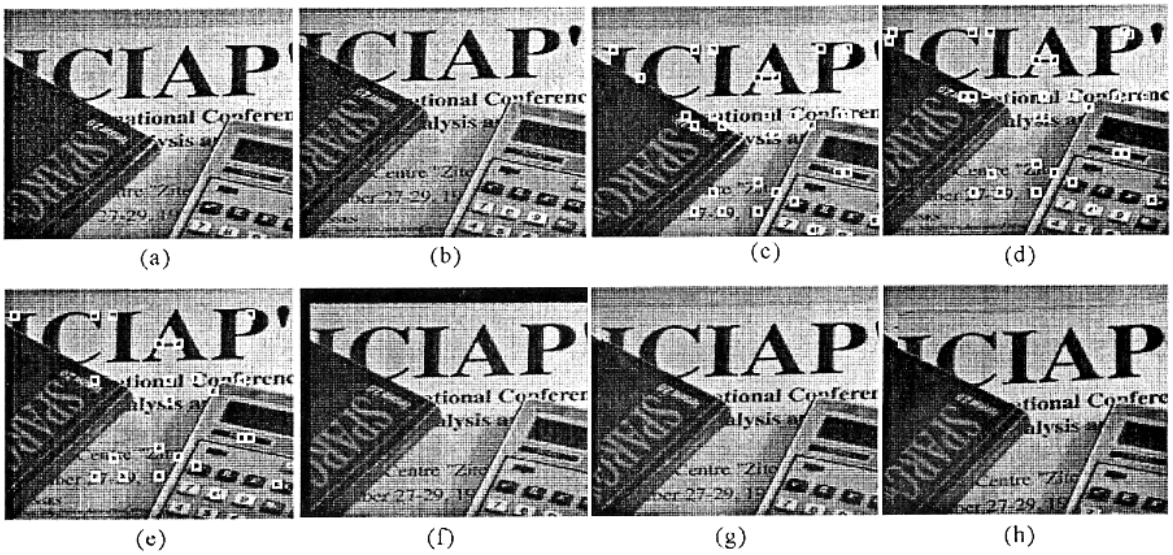

图 2 稳像算法的仿真结果

Fig.2 Simulation results of stabilization algorithm

另一方面,图像序列稳像效果的评价与比较是很困难的,因为图像序列的理想运动无从得知。因此,利用观察者的感观评价是一个很好的选择。基于这点考虑,本文选用德国 Dynapel 公司推出的稳像软件‘ Steady Hand ”与本文的稳像算法处理同一段抖动视频,然后对比两者的稳定效果。结果表明,本文算法的稳像效果明显优于‘ Steady Hand ”,这是因为算法设计时充分考虑了稳像结果的鲁棒性。

参 考 文 献:

[1] Litvin A, Konrad J,et al. Probabilistic Video Stabilization Using Kalman Filtering and Mosaicking [A]. IS&T/SPIE Symposium on Electronic Imaging, Image and Video Communications and Proc.[C], 2003,5022:663 – 674.
[2] Tomasi C,Kanade T. Detection and Tracking of Point Features[R]. Carnegic Mellon University Technical Report CMU-CS-91-132, 1991.
[3] Shi J B,Tomasi C. Good Features to Track [J]. IEEE Conference on Computer Vision and Pattern Recognition, CVPR,1994:593 – 600.