基于 Agent 的分布仿真中层次时间管理机制的研究与实现*)

叶超群　吴　集　金士尧

(国防科学技术大学并行与分布处理国家重点实验室　长沙 410073)

摘　要　基于 Agent 的分布仿真是当前国内外研究的热点,而时间管理是其中的关键技术,也是影响系统性能的主要因素。在大规模的基于 Agent 的分布仿真中,数目巨大的 Agent 给时间管理带来了挑战,本文提出了一种层次的时间管理机制,它采用了局部仿真引擎和全局仿真引擎来进行时间管理,通过增加结点内的计算开销来提高时间管理中的并发性、减少网络传送开销,从而提高仿真的性能。最后对该机制进行了性能评测。

关键词　Agent,分布仿真,时间管理

Study and Implementation of a Hierarchical Time Management Mechanism in Agent-Based Distributed Simulation

YE Chao-Qun　WU Ji　JIN Shi-Yao

(National Laboratory for Parallel & Distributed Processing, National University of Defense Technology, Changsha 410073)

Abstract　Agent-based distributed simulation is an active research area currently. Among this area, time management is not only one of the most important technologies but also the key factor affecting system performance. However, there are a great number of Agents which challenges the efficiency of time management. In this paper, a hierarchical time management mechanism is proposed and it is composed of Local Simulation Engines and Global Simulation Engine. It improves parallelism in time management and reduces network communication overhead by increasing local computation overhead in each node, which helps to enhance the performance of simulation. Lastly, the performance of this mechanism is analyzed and tested.

Keywords　Agent, Distributed simulation, Time management

1　引言

基于 Agent 的仿真是当前国内外研究的热点,被广泛应用于许多重要的领域,例如生态系统、经济、政治、军事和社会等,通常这些领域都很复杂,系统的行为、特性难以进行分析和验证,计算机仿真是重要的研究手段[1~5]。

在基于 Agent 的仿真中,目标系统由自底向上建立的多 Agent 系统模型来建模,仿真中往往存在数目巨大的 Agent,并且每个 Agent 自身就是一个复杂的计算系统,仿真所需的计算量远远超出了单处理机的计算能力,通常采用分布仿真技术在网络多机环境中进行计算,来提高仿真的速度[2, 6]。时间管理是分布仿真中的关键技术,也是影响分布仿真性能的主要因素,其本质是依据仿真中事件(消息)发生的逻辑时间对事件进行排序,从而保证所有的 Agent 能够在适当的时间以相同的视图观察到这些事件的发生,并以符合因果约束关系的顺序接收和处理这些事件。为了能够做出正确的决策,时间管理需要知道所有 Agent 的当前状态,因此如何高效地实现时间管理是提高仿真性能的关键。

在本文中,针对基于 Agent 的仿真中存在着数目巨大的实体的特点,结合基于 Agent 的分布仿真框架,提出并实现了一种层次的时间管理机制,该机制通过增加结点内局部计算开销来提高时间管理中的并发性、减少网络传送开销,从而提高仿真的性能。在本文的以下章节中,首先介绍了相关工作以及基本概念,然后结合基于 Agent 的分布仿真框架,提出了层次的时间管理机制,并对该时间管理机制的性能进行了分析和评测,最后进行了总结。

2　相关研究

2.1　基于 Agent 的分布仿真平台

目前国际上已经有了多种基于 Agent 的分布仿真平台,其中具有代表性有美国 Massachusetts 大学开发的仿真平台 Farm[7],Carnegie Mellon 大学的仿真中间件 MPADES[8, 9] 以及加拿大 Manitoba 大学智能 Agent 实验室的仿真平台 DGensim[10] 等,它们把系统中的 Agent 分布到多机上去并发执行来提高仿真的速度,每个处理机结点运行单个或者多个 Agent。在这些基于 Agent 的分布仿真平台中,大多采用了时间步进的方式来推进 Agent 及其环境的时钟,它们的时间管理机制是集中式的,即共享的环境运行在一个单独的处理机上,该结点上的仿真引擎完成系统的时间管理,因此仍然无法满足大规模的基于 Agent 的分布仿真的需要。

2.2　HLA/RTI 中的时间管理

HLA 作为分布式仿真领域的标准,代表着当今分布式仿真的主流技术。提出 HLA 标准的目的在于解决不同仿真系统的互联和互操作问题,它能够同时支持保守和乐观的两种逻辑时间同步推进机制以及实时仿真,并支持这些机制的协同推进,这增加了时间管理服务的复杂性,使其成为 RTI 实现中的难点问题之一[11, 12]。到目前为止,国内外仅有少数几个 RTI 软件完整实现了时间管理服务[13~15],它们的结构可以分为集中式和分布式两种,但这些 RTI 的实现只能够应用于小规模的仿真应用,无法适应大规模仿真的需要。

3　一种层次的时间管理机制

3.1　基本思想

时间管理的核心算法是计算最大可用逻辑时间 GALT (Greatest Available Logical Time),它是 Agent 可以安全推进到的逻辑时间上限,当 $Agent_i$ 请求推进到的逻辑时间 T 时,如果 T 不大于它的 $GALT_i$,则 $Agent_i$ 可被授予时间 T,其他 Agent 不会因此而接收到错误时序的消息。GALT 的定义如

*)本课题受国防预研基金资助。叶超群　博士生,主要研究方向为基于 Agent 的分布仿真;吴　集　博士生,研究方向为基于 Agent 的仿真建模;金士尧　博士生导师,主要研究方向为分布计算与仿真、性能评价等。

下：

定义 1 最大可用逻辑时间 GALT：$Agent_i$ 的最大可用逻辑时间 $GALT_i$ 指的是 $Agent_i$ 可以安全推进到的最大逻辑时间。

GALT 的计算与前瞻值 Lookahead 以及输出时间 OLT (Output Logical Time)密切相关，它们的定义分别为：

定义 2 前瞻值 Lookahead：如果 $Agent_i$ 在逻辑时间 T 仅能够发送时戳值不小于 $T+L$ 的消息，则 L 就是 $Agent_i$ 的前瞻值 $Lookahead_i$ 的值。

定义 3 输出逻辑时间 OLT：$Agent_i$ 的输出逻辑时间 OLT_i 指的是 $Agent_i$ 能够发送的事件(消息)的最小逻辑时间。

式(1)和(2)给出了计算 $GALT_i$ 的方法，其中 A 为仿真中所有 Agent 的集合(环境可以看成是一个特殊的 Agent)，T_i 为 $Agent_i$ 当前的逻辑时间：

$$OLT_i = T_i + Lookahead_i \tag{1}$$

$$GALT_i = \{Min(OLT_j) \mid j \in A, i \neq j\} \tag{2}$$

可以看出，Agent 的 GALT 等于除了它自身之外的其他所有 Agent 的 OLT 中的最小值，假设将集合 A 中的所有 Agent 按照其输出时间从小到大进行排序，得到的有序序列为 $\langle Agent_0, Agent_1, Agent_2, \cdots, Agent_N \rangle$，则 $GALT_0 = OLT_1$，而 $GALT_i = OLT_0 (i \in [1, N])$。因此 $GALT$ 的计算也可以表示为公式(3)，它只与 GALT 的最小值以及次小值有关，其中 $OLT_{minimum}$ 和 OLT_{minor} 分别为集合 A 中所有 Agent 的 OLT 中的最小值和次小值：

$$GALT_i = \begin{cases} OLT_{minor} & \text{if } OLT_i = OLT_{minimum} \\ OLT_{minimum} & \text{if } OLT_i \neq OLT_{minimum} \end{cases} \tag{3}$$

这样 GALT 的计算就归结为求解 OLT 中的全局最小值和次小值，而在基于 Agent 的分布仿真框架中，每个处理机结点上运行有单个或者多个 Agent，因此可以采用层次的时间管理机制来提高效率，通过局部仿真引擎分别计算每个处理机结点上 OLT 中的局部最小值和次小值，再由全局仿真引擎综合这些信息计算 OLT 中的全局最小值和次小值，最后局部仿真引擎根据计算结果完成本结点上 Agent 的时间推进。

3.2 局部仿真引擎(Local Simulation Engine，LSE)

在每个处理机结点上有且仅有一个局部仿真引擎 LSE，当参与仿真的 Agent 向 LSE 注册时，LSE 为该 Agent 创建一个代理，加入代理链表。代理维护有与 Agent 的时间管理相关的信息，主要包括：

- Agent 引用：通过该引用可以查找到被代理的 Agent；
- Agent 的状态：时间管理赋予 Agent 两种状态。当 Agent 发出时间推进请求，而该请求未被授予时，Agent 处于未授予状态，而如果 Agent 的时间请求已经被授予，而又未发出新的时间推进请求，则 Agent 处于已授予状态；
- 预期推进时间：预期推进时间指的是当 Agent 处于未授予状态时，未被授予的请求所期待推进到的逻辑时间；
- 可被授予时间：可被授予时间指的是 Agent 处于授予状态时，最近一次被授予的逻辑时间；
- 消息队列：当其他 Agent 通过通讯系统发送消息给被代理的 Agent 时，所有的消息都被缓存在代理的消息队列中，由时间管理机制在适当的时机提交给 Agent 进行处理；

当 Agent 发送时间推进请求进入未授予状态时，LSE 将其代理加入等待队列等待处理。LSE 采用一个单独的线程进行驱动，它执行下面的算法完成时间管理：

```
while (run_f){
    if(! waitingAgents. isEmpty()){ //是否有 Agent 处于未授予状态
        if(recalculatelocalCriteria()){ //重新计算局部最小和次小输出时间
            //如果局部最小或者次小输出时间发生了改变，则把最新的计算结果发送给全局引擎，用于计算全局的最小和次小输出时间
            notifyGlobalEngine ( localMinimumTS, localMinorTS);
        }
        updateGlobalCriteria(); //更新全局最小、次小输出时间
        doAdvanceTime(); //完成时间授予，提交缓存的消息
    }
    ……
    Thread. yield(); //主动放弃占用 CPU
}
```

算法中，重新计算 OLT 的局部最小值和次小值需要遍历 LSE 的代理链表，它不涉及其他结点上 Agent 的状态，可以和其他处理机结点的操作并发执行。当 OLT 的局部最小值或者次小值发生变化时，LSE 向全局仿真引擎 GSE 发送最新计算结果，并根据 GSE 计算得到的 OLT 的最新全局最小值和次小值，分别计算每个处于未授予状态的 Agent(可以通过 LSE 的等待队列查找)的 GALT，如果 Agent 的时间推进请求能够被授予，则 LSE 将其代理从等待队列中取出，将其状态置为授予状态，并且把缓存在其代理消息队列中的时戳值小于可授予时间的消息提交到 Agent 的事件队列中。

3.3 全局仿真引擎(Global Simulation Engine，GSE)

在整个分布仿真系统中，有且仅有一个全局仿真引擎 GSE，当局部仿真引擎 LSE 向 GSE 注册时，GSE 以 LSE 所在的处理机的地址为索引，在哈希表中建立 LSE 的状态信息，保存该处理机结点上 OLT 的局部最小值和次小值。当有 LSE 通知 GSE 自己的 OLT 局部最小值和次小值改变了时，GSE 更新其对应的状态信息，并且遍历所有 LSE 的状态信息，重新计算 OLT 的全局最小值和次小值，并将最新的计算结果广播给所有的 LSE。GSE 的执行受 LSE 更新消息的触发，执行如下算法：

```
updateLocalCriteria(localCriteria); //更新 LSE 对应的状态信息
if(recalculateGlobalCriteria()){ //重新计算 OLT 的全局最小值和次小值
    notifyAllEngine(globalCriteria()); //将最新的计算结果广播给所有的 LSE
}
```

3.4 基于 Agent 的分布仿真框架

在采用了层次的时间管理机制后，基于 Agent 的分布仿真框架如图 1 所示，每个处理机结点上运行单个或者多个 Agent，通讯系统、仿真引擎和 Agent 支持库共同构成了 Agent 的运行支撑环境。其中，通讯系统提供了可靠的结点间通讯服务，LSE 与 GSE 之间的通讯通过调用通讯接口完成，并且通讯系统提供了高效、透明的消息传递服务，使得 Agent 之间进行消息交互时，消息的发送方不需要知道接收方在哪个处理机结点上。Agent 支持库提供了 Agent 建模的基本框架，用户通过继承支持库中的 Agent 模板，可以快速开发出适合于应用需要的 Agent。

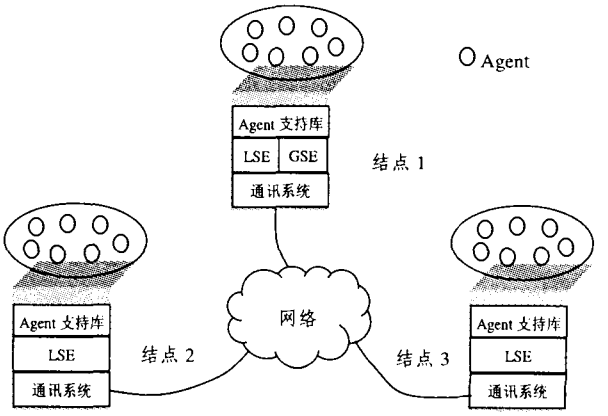

图 1 基于 Agent 的分布仿真框架

Agent 通过下面的方式使用时间管理服务：

```
While(run_f){
    advanceTimeRequest(nextTime); // 发送时间推进请求
    while(! advanceTimeGranted()){ //等待时间推进请求被授予
      sleep(interval);
    }
    AT = nextTime; // Agent 推进自己的逻辑时钟
    while (! eventQueue.isEmpty()){ //查看事件队列是否为空
      evt = (Event)eventQueue.removeFirst(); //从事件队列中取出时戳最小的事件
      process(evt); //对事件进行处理
    ......
    }
}
```

4 性能评测

4.1 开销分析

在层次的时间管理机制中，其性能开销分别由 LSE 计算 OLT 的局部最小值和次小值、GSE 计算 OLT 的全局最小值和次小值、LSE 完成时间推进以及网络通讯等开销组成，可以表示为：$t_{TM} = t_{LSE} + t_{GSE} + t_{advanceTime} + 2t_{network}$。假设在分布仿真环境中处理机结点的数目为 N，每个处理机结点上的 Agent 数目为 M，网络通讯一次(包括广播)的最大开销为 K，则 LSE 计算 OLT 的局部最小值和次小值需要遍历代理链表，$O(t_{LSE})=O(M)$；GSE 计算 OLT 的全局最小值和次小值需要遍历 LSE 链表，$O(t_{GSE})=O(N)$；LSE 完成时间推进需要遍历等待队列，$O(t_{advanceTime})=O(M)$；K 为常数。可以得出

$$O(t_{TM})=O(M)+O(N)$$
$$=O(MN/M)+O(M)=O(MN/N)+O(N) \quad (4)$$

因此，当 Agent 的总数 MN 为常数时，时间管理的开销与处理机结点的数目 N 或者单个结点上的 Agent 数目 M 之间是非线性关系。

4.2 性能测试

我们在基于 Agent 的分布仿真平台 JCass[6] 上实现了层次的步进时间管理机制，测试了步进一步的时间管理开销，测试环境由 8 台 PC 机通过 100M 以太网互联组成，JDK 的版本为 1.41_02。

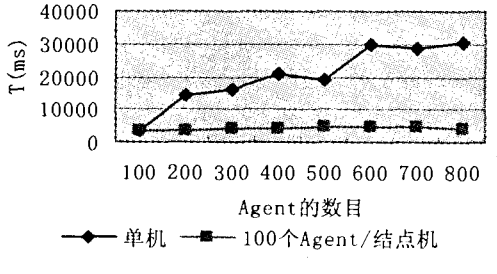

图 2 不同规模的时间管理开销比较

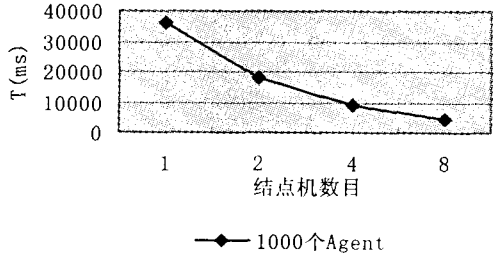

图 3 不同结点机的时间管理开销

测试 1 比较了不同仿真规模下的时间管理开销，结果如图 2 所示。可以看出在单机环境下，随着 Agent 数目的增加，时间管理开销增加较快，而通过增加处理机结点，每个结点上运行 100 个 Agent 时，随着 Agent 数目的增加时间管理的开销几乎没有变化。

测试 2 对相同仿真规模、不同数目处理机结点的情况下的时间管理开销进行了比较，结果如图 3 所示，可以看出在系统规模为 1000 个 Agent 不变时，随着处理机结点的增加，时间管理的开销明显减少。

测试结果表明，层次的时间管理机制具有较好的扩展性，基本能满足大规模基于 Agent 的仿真的需要。

结束语 基于 Agent 的仿真受到了广泛的关注，它提供了解决复杂问题的新方法。为了能够支持大规模的基于 Agent 的仿真，需要在网络多机环境中进行分布仿真，而时间管理是其中的关键技术，也是影响系统性能的主要因素。本文提出了一种层次的时间管理机制，它通过局部仿真引擎 LSE 和全局仿真引擎 GSE 协同完成时间管理，提高了时间管理中的并发性，并减少了网络通讯的开销。该方案被应用于基于 Agent 的复杂系统分布仿真平台 JCass 中，测试表明该时间管理机制具有较好的扩展性，基本能够满足大规模的基于 Agent 的分布仿真的需要。

参 考 文 献

1 http://www.swarm.org.
2 罗批，司光亚，胡晓峰，杨镜宇. 基于 Agent 的复杂系统建模仿真方法研究进展. 装备指挥技术学院学报，2003,14(1):1～5
3 周佩玲，夏懿，李立文，虚拟股市的建模与仿真. 计算机仿真，2002,19(6):69～74
4 Holland J H. Emergence: From Chaos to Order. Addison-Wesley Publishing Company, Inc, 1998
5 http://www.santafe.edu.
6 李宏亮. 基于 Agent 的复杂系统分布仿真. 国防科学技术大学研究生院. 2001
7 Horling B, Mailler R, Lesser V. Farm: A Scalable Environment for Multi-Agent Development and Evaluation. In: Proc. of the 2nd Intl. Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2003). May 2003
8 Riley P, Riley G. SPADES - A Distributed Agent Simulation Environment with Software-in-the-Loop Execution. in Winter Simulation Conference Proceedings, 2003
9 Riley P. MPADES: Middleware for Parallel Agent Discrete Event Simulation. in RoboCup-2002: Robot Soccer World Cup VI. 2003. Berlin, Germany: Springer Verlag.
10 Anderson J. A Generic Distributed Simulation System for Intelligent Agent Design and Evaluation. In: Proc. of 10th Conf. on AI Simulation and Planning in High Autonomy Systems(AIS-2000). March 2000.
11 HLA Time Management: Design Document. Version 1.0. Defense Modeling and Simulation OfficeDefense Modeling and Simulation Office. 1996
12 Fujimoto R M. Time Management in the High Level Architecture. Simulation, 1998,71(6):388～400
13 High Level Architecture Run-Time Infrastructure RTI 1.3-Next Generation Programmer's Guide Version 3. Department of Defense, Defense Modeling and Simulation Office. 1999.9
14 刘步权. 分布式仿真运行支撑平台中时间管理服务的研究. 国防科学技术大学研究生院，2004
15 姚益平. 高性能分布式交互仿真运行支撑平台关键技术研究. 国防科学技术大学研究生院，2003