ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.6, June 2010, pp.1208−1219
doi: 10.3724/SP.J.1001.2010.03630

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

分布式交互仿真中的异步时钟一致性控制方法*

张 伟[1+], 周航军[1], 彭宇行[1], 李思昆[2]

[1](国防科学技术大学 并行与分布处理国家重点实验室,湖南 长沙 410073)

[2](国防科学技术大学 计算机学院,湖南 长沙 410073)

Asynchronous Time Consistency Control Methods in Distributed Interactive Simulation

ZHANG Wei[1+], ZHOU Hang-Jun[1], PENG Yu-Xing[1], LI Si-Kun[2]

[1](National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073, China)

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

+ Corresponding author: E-mail: zw_nudt@163.com

Zhang W, Zhou HJ, Peng YX, Li SK. Asynchronous time consistency control methods in distributed interactive simulation. *Journal of Software*, 2010,21(6):1208−1219. http://www.jos.org.cn/1000-9825/3630.htm

Abstract: This paper proposes a model called asynchronous time consistency (ATC) based on call-back clock method in distributed interactive simulation system (DIS). This model utilizes the asynchronism of different nodes' clocks to re-distribute the time resources and improve the overall performance without compromising the system functionality. With this model, the paper firstly designs a fast approximation method based on the global delay information, and proves the effectiveness of the algorithm; secondly, according to the actual network condition, it also designs a fast iteration method based on the partial delay information. In the end, experiments are conducted and analyze to static and dynamic performance of these methods. Experimental results show that the ATC methods can improve the interaction performance effectively.

Key words: distributed interactive simulation; continuous model; asynchronous time; consistency; interaction efficiency

摘 要: 提出了在分布式交互仿真中基于回拨时钟的异步时钟一致性控制模型,利用节点间时钟的异步性对系统时间资源进行重新分配,达到了在保证原有系统功能正确性的前提下有效提高系统整体性能的目的.根据上述模型,首先提出了基于全局信息的快速拟合法,并证明了该算法的有效性;之后又根据实际网络情况提出了基于局部延迟信息的逐步逼近法;最后通过实验分析了上述方法的静态和动态性能.实验结果表明,异步时钟方法对系统的交互性能有显著提高.

关键词: 分布式交互仿真;连续模型;异步时钟;一致性;交互效率

中图法分类号: TP391 文献标识码: A

分布式交互仿真(distributed interactive simulation,简称 DIS)[1]是一种基于分布式仿真技术的实时网络交互

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z332 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2005CB321801 (国家重点基础研究发展计划(973))

Received 2008-12-04; Revised 2009-01-14; Accepted 2009-04-13

环境,它通过分布在各地的计算节点之间的通信,共享交互环境的局部或全局数据信息,协同完成从分布到多计算节点上的任务,给用户提供一种整体的、真实的、可沉浸的虚拟空间.分布式交互仿真技术由于其扩展性好、适应性广等特点,近年来发展很快,并已在军事仿真、网络娱乐、远程教育、工程设计以及电子商务等方面得到了广泛的应用.

一致性问题是分布式交互仿真中的本质和核心问题.由于存在着网络传输延迟,各节点间的消息通信不能实时到达,会产生消息序错乱以及时序关系混乱等情况,进而导致理解歧义、违背期望、因果颠倒[2]等问题,严重影响交互仿真的正常运行.并且,一致性问题还会影响系统的公平性和正确性[3],使得最终仿真得到的结果不可信.因此,能否有效解决交互仿真中各节点间的一致性问题,已经成为了制约 DIS 系统进一步发展的关键要素.

一般说来,分布式交互仿真系统的状态空间都比较庞大,要直接维护状态间的一致性需要较大的时间空间开销.因此,当各节点的初始系统状态保持一致时,只要保证改变系统状态的要素在各节点正确执行即可.对于离散模型来说,由于状态的改变仅由事件产生,因此只需要维护各节点按照同样的顺序处理事件.这一类问题已经得到了较深入的研究,如文献[4–6].而对于连续模型来说,由于系统状态还伴随着时间的改变而不断变化,因此需要保证各事件在同样的时间点和各节点上得以执行.

现有的维护连续模型的一致性控制方法都存在着明显的不足.如文献[7]提出的锁步进同步技术,它通过强制阻塞节点的时钟推进,直至所有的节点完成该时间步的计算后才进入下一个时间周期,从而避免了不一致现象的发生.该方法主要研究如何从功能上保证系统的正确性要求,而在性能方面则完全无法满足交互仿真的基本要求.为了改善上述方法的性能缺陷,文献[8]提出了基于间距一致的延迟一致性,通过只延迟异地事件而不延迟本地事件的方法来提高系统的性能,在一定程度上改善了系统的响应能力.然而事实上,该方法是以牺牲系统功能的方法来达到提高性能的目的,在很多情况下,该方法的控制结果无法满足消息序的一致性,进一步导致各种不一致现象的发生.而文献[9]则认为,连续模型的一致性控制是针对 DIS 系统功能和性能的折衷,针对一方面的维护一定会导致另一方面能力的下降.

针对目前 DIS 系统中一致性控制的功能和性能无法同时满足的问题,本文通过对连续模型一致性控制过程的深入分析,提出一种基于回拨时钟的异步时钟一致性控制模型,达到了在不改变 DIS 系统功能正确性的基础上有效改善系统性能的目的.其基本思想是,通过放松节点间的仿真时间同步约束条件对系统时间资源进行重新调配,均匀化节点间消息延迟的关键路径,从而达到在保证 DIS 系统原有功能性的同时大大提高系统性能的目的,为分布式交互仿真在大规模网络条件下的应用提供支持.

本文首先给出上述方法的计算过程,然后在理论上证明该方法的能力,最后通过实验结果验证方法的有效性.

1 问题描述

设 V 代表全体仿真节点的集合,O 代表系统中产生的全体事件的集合,对于任意的 $v_i\in V$,G_i 代表节点 v_i 产生的事件集合,R_i 代表节点 v_i 能够收到的事件集合.令 $E_i=G_i\cup R_i$,称 E_i 为节点 v_i 可感知的事件集合.对于任意的 $o_m\in O$,若 $o_m\in G_i$,则 $TG_i(o_m)$代表事件 o_m 在节点 v_i 上的产生时间,$TS(o_m)$代表节点 v_i 设定的事件 o_m 在各个节点的执行时间;若 $o_m\in R_i$,则 $TR_i(o_m)$代表事件 o_m 在节点 v_i 上的接收时间;若 $o_m\in E_i$,$TE_i(o_m)$代表事件 o_m 在节点 v_i 上实际的执行时间.

分布式交互仿真中的功能正确性有 3 个层次的一致性需求,即序一致、间距一致和执行时间一致,表示为:

定义 **1**(序一致).

$$\forall v_i,v_j\in V;o_m,o_n\in E_i\cap E_j \quad \text{s.t.} \quad TE_i(o_m)\le TE_i(o_n)\rightarrow TE_j(o_m)\le TE_j(o_n).$$

序一致是指对于任意两个事件 o_m 和 o_n,若它们均能为节点 v_i 和 v_j 所感知,则它们在节点 v_i 和 v_j 上的执行顺序一致.

定义 **2**(间距一致).

$$\forall v_i,v_j\in V;o_m,o_n\in E_i\cap E_j \quad \text{s.t.} \quad TE_i(o_m)-TE_i(o_n)=TE_j(o_m)-TE_j(o_n).$$

间距一致比序一致更进一步,它在保持执行顺序一致的基础上还要求事件 o_m 和 o_n 在节点 v_i 和 v_j 上执行的时间间距保持一致.

定义 **3**(执行时间一致).

$$\forall v_i,v_j\in V;o_m\in E_i\cap E_j \quad \text{s.t.} \quad TE_i(o_m)=TE_j(o_m).$$

执行时间一致的要求最为严格,它要求所有的事件在各个节点上同时得到执行.

对于 DIS 中的离散模型,只要保证序一致即可满足系统正确性要求,而对于连续模型,一般要求达到执行时间一致,少数应用只需要达到间距一致即可[8].

除了系统的功能正确性要求外,还要考虑系统的性能要求,即响应时间的要求,它代表了系统对交互的响应能力.下面给出 3 种粒度的响应时间定义,即事件的响应时间、节点的响应时间和系统的响应时间.

设事件 o_m 是节点 v_i 上产生的事件,令 $CD_i(o_m)$为事件 o_m 在节点 v_i 的本地延迟时间,其值等于事件 o_m 的设定执行时间 $TS(o_m)$和产生时间 $TG_i(o_m)$之差.令 $CF_i(o_m)$代表事件 o_m 在节点 v_i 上的响应时间,其值等于事件 o_m 在节点 v_i 的本地延迟时间 $CD_i(o_m)$,即有下面的定义.

定义 **4**(事件的响应时间).

$$CF_i(o_m)=CD_i(o_m)=TS(o_m)-TG_i(o_m),v_i\in V,o_m\in G_i.$$

对于系统中的各节点来说,节点的响应时间定义为其所有产生事件的响应时间平均值,即

定义 **5**(节点的响应时间).

$$CF_i=\frac{\sum_{\forall o_m\in G_i}CF_i(o_m)}{|G_i|},v_i\in V.$$

系统的响应时间定义为各节点响应时间的平均值,即有下面的定义.

定义 **6**(系统的响应时间).

$$CF_{all}=\frac{\sum_{\forall v_i\in V}CF_i}{|V|}.$$

如果 DIS 系统的功能要求达到定义 3 所描述的执行时间一致,并且系统的响应时间为 0,则称系统达到了绝对一致性(absolute consistency)[8].然而,绝对一致性只是一种理想的情况,由于网络延迟不可忽略,实际 DIS 系统中无法达到绝对一致性.因此,本文以最小化系统响应时间为目标,并设定如下约束条件:

约束条件 **0**(一致性功能及性能约束).

$$\forall v_i,v_j\in V;o_m\in E_i\cap E_j \quad \text{s.t.} \quad TE_i(o_m)=TE_j(o_m),$$
$$\forall v_i\in V \quad \text{s.t.} \quad CF_i\le CFM_i,$$

其中,CFM_i 为节点 v_i 的最大容忍响应时间.因此,约束条件 0 表示本文的目标是在保证定义 3 所述的执行时间一致性以及保证各节点的响应时间不超过最大容忍响应时间的前提下,尽可能地减小系统的响应时间.

2 基于回拨时钟的异步时钟一致性模型

2.1 DIS中的异步时钟

约束条件 **1**(执行时间一致性约束)[8]. 要达到定义 3 中描述的执行时间一致,需要保证事件 o_m 在任意接收节点上被设定的执行时间都不小于其接收时间,即 $TS(o_m)\ge TR_i(o_m),v_i\in V,o_m\in R_i$.

在各节点时钟同步的情况下,要满足约束条件 1,事件 o_m 在节点 v_i 上产生后要在本地延迟一段时间,其值取决于节点 v_i 和所有接受事件 o_m 的节点间通信延迟的最大值,即

约束条件 **2**(同步时钟的本地延迟约束).

$$CD_i(o_m)\ge \operatorname{Max}_{\forall v_j\in V,o_m\in R_j} D_i^j,v_i\in V,o_m\in G_i,$$

其中,$CD_i(o_m)$代表事件 o_m 在节点 v_i 上的本地延迟时间, D_i^j 代表节点 v_i 和节点 v_j 之间的消息通信延迟.

在实际系统中,由于互联网络上各节点之间通信和计算能力存在差异,可能由于某些节点的通信延迟和计算延迟较大从而影响了整个系统的响应能力,如图 1 所示.

在图 1 中,存在着 *A*,*B*,*C* 这 3 个仿真节点,为简化起见,我们假设节点间通信延迟是对称的,其中,*A*,*B* 间通信延迟为 2,*A*,*B* 两节点和 *C* 之间通信延迟都为 10.则根据约束条件 2 可知,3 个节点的本地延迟时间均为 10.虽然 *A*,*B* 之间的通信延迟为 2,但由于它们和 *C* 之间的延迟均为 10,为达到定义 3 所描述的执行时间一致的条件,*A* 发送给 *B* 的消息必须要在 *B* 上延迟 8 后才能执行,从而浪费了系统有限的时间资源.

在现有方法的认识中,分布式交互仿真系统由于存在着实时的交互,仿真时间需要和外部的墙钟时间严格同步.因此,同时执行的概念就需要所有的事件在物理世界的墙钟时间上同时执行,这就决定了响应时间受限于网络传输延迟的最大值,也造成了系统一致性功能和性能之间不可调和的矛盾.实际上,仿真时间的严格同步是一种人为简化的假设,在这种强制的假设前提下,一致性控制方法可以比较容易地进行时序控制,但是因此导致的功能性能之间的尖锐矛盾就无法得到有效的解决.因此,本文从 DIS 系统一致性需求的本质出发,采用基于异步时钟的一致性控制模型,令各仿真节点的仿真时间在相对时间轴上偏差一小段距离,进而研究其对一致性控制效果的影响.

下面举例说明异步一致性控制方法对控制效果带来的影响,如图 2 所示.在图 2 中,存在着两个节点 v_1,v_2.节点 v_1 在 T 时刻向节点 v_2 发送消息 E_1,该消息的延迟时间为ΔT;v_2 节点应该在 $T+\Delta T$ 时刻收到该事件,因此 v_1 也必须等待ΔT时刻才能执行该事件.因此,节点 v_1的响应时间为ΔT.此时,若节点 v_2将自己的时钟回拨时间ΔT,称为节点 v_2 的回拨时钟值,则节点 v_1 发送事件 E_1 时节点 v_2 的时钟为 $T-\Delta T$,节点 v_2 收到该事件的时刻为 T.因此,节点 v_1 发送完事件 E_1 后即可立即执行该事件,其响应时间就变为 0.但是,若节点 v_2 向节点 v_1 发送消息,则需要等待更长的时间来保证定义 3 的执行时间一致性,这种等待约束的变化需要引入异步时钟的本地延迟约束来加以计算.

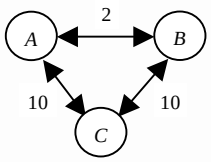


Fig.1 Impact of one-node delay to the system responsiveness

图 1 单节点延迟对系统响应能力的影响

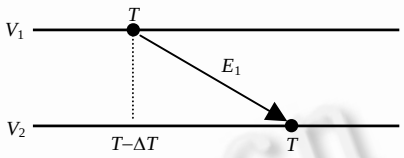


Fig.2 Diagram of call-back clock method

图 2 回拨时钟方法示意图

2.2 异步时钟的本地延迟约束

在各仿真节点的仿真时钟不同步的情况下,原有的本地延迟约束条件无法满足定义 3 的执行时间一致,因此需要定义新的约束条件.在假定各节点回拨时钟值已确定的前提下,基于异步时钟的本地延迟约束既与节点间的通信延迟有关,也与收发节点的回拨时钟值有关,其定义如下:

约束条件 **3**(异步时钟的本地延迟约束).

$$CD_i(o_m) \geq \underset{\forall v_j \in V, o_m \in R_j}{\text{Max}} (D_i^j + \Delta T_i - \Delta T_j), v_i \in V, o_m \in G_i,$$

其中,ΔT_i,ΔT_j 代表节点 v_i,v_j 的回拨时钟值.

定理 **1**. 在异步时钟条件下,约束条件 3 满足定义 3.

证明:任取系统中某事件 o_m,其发送节点为 v_i,任取 o_m 的接收节点 v_k.假设系统标准的仿真时间为 T_s,则节点 v_i,v_k 的仿真时间分别为 $T_s-\Delta T_i$,$T_s-\Delta T_k$,由约束条件 3 可知:

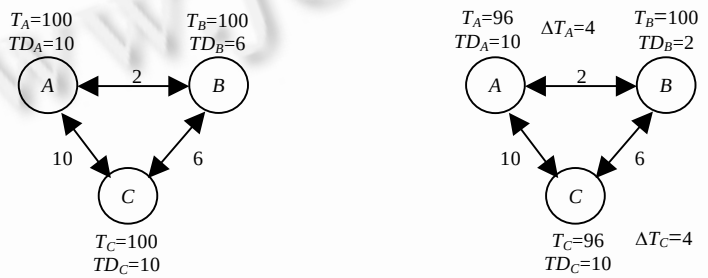
$$CD_i(o_m) \geq \underset{\forall v_j \in V, o_m \in R_j}{\text{Max}} (D_i^j + \Delta T_i - \Delta T_j) \geq D_i^k + \Delta T_i - \Delta T_k .$$

而又由定义 4 可知 $TS(o_m) = TG_i(o_m) + CD_i(o_m) \geq TG_i(o_m) + D_i^k + \Delta T_i - \Delta T_k$,假定节点 v_i 在 $T_s-\Delta T_i$ 时刻产生事件,则此时节点 v_k 的仿真时间为 $T_s-\Delta T_k$,在节点 v_k 接收事件 o_m 的时间为 $TR_k(o_m) = D_i^k + T_s - \Delta T_k$,而在节点 v_k 上

执行事件 o_m 的时间为 $TS(o_m) \geq TG_i(o_m) + D_i^k + \Delta T_i - \Delta T_k = T_s - \Delta T_i + D_i^k + \Delta T_i - \Delta T_k = TR_k(o_m)$,满足约束条件 1 所述需求,故满足定义 3 所述执行时间一致.证毕. □

由定理 1 可知,在满足约束条件 3 的情况下可以达到 DIS 系统的一致性功能需求,而且通过对时钟资源的重新分配可以达到提高系统整体响应能力的目的.下面首先举例说明如何通过回拨时钟来减小系统的响应时间,下一节再给出方法有效性的形式化证明.

在图 3(a)中,*A*,*B*,*C* 这 3 个节点相互间的传输延迟分别为 2,6,10,假设各节点仿真时钟均为 100.由于采用同步时钟机制,因此为达到各节点执行时间一致的需求,*A*,*C* 两节点的本地延迟时间要不小于 10,*B* 节点的本地延迟时间不小于 6.也就是说,对于 *B* 节点来说,现在产生的事件必须加盖不小于仿真时间 106 的时间戳,才能保证该事件在各个节点上同时执行.而对于异步时钟来说,由于各节点间的仿真时钟不是完全同步,因此可以对某些节点的时钟值进行回拨处理.如图 3(b)所示,我们设置 *A*,*C* 两节点的回拨时钟值为 4,则两节点当前的仿真时间就变为 96.根据约束条件 3,可知 *A*,*C* 两节点的本地延迟时间不变,仍为 10,而 *B* 节点的延迟时间减小为 2.也就是说,当前 *B* 节点可以产生时戳为 102 的事件,在此条件下执行时间一致的需求仍然得到满足.可见,在异步时钟策略的前提下,通过系统时间资源的有效调配,可以在不改变功能需求的前提下提高系统的整体性能,达到改善 DIS 系统交互能力的目的.



(a) Synchronous time model (a) 同步时钟模型

(b) Asynchronous time model (b) 异步时钟模型

Fig.3 Delay

图 3 延迟

3 回拨时钟值的确定

3.1 模型分析

设置合适的回拨时钟值是保证异步时钟方法达到整体性能最优化的关键要素,因此本节讨论如何确定各个节点的回拨时钟值.

设系统中存在 *n* 个节点,各节点间通信延迟构成矩阵:

$$X = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix},$$

其中,a_{ij} 代表节点 v_i 和节点 v_j 之间的通信延迟,因此可得 a_{ii}=0,i=1,…,n.若假设节点间通信延迟是对称的,则可知 a_{ij}=a_{ji},i,j=1,…,n.

系统整体的响应时间可通过计算该矩阵的行范数和来得到,即

$$f(X) = \sum_{i=1}^{n} \| a_{ij} \|_{\infty} \ (\| a_{ij} \|_{\infty} = \max\{a_{ij}\}, j = 1,...,n),$$

而针对该矩阵的回拨时钟值可构成向量:S=(s_1,s_2,…,s_n),其中,s_i 代表节点 i 的回拨时钟值.通过此回拨时钟向量,可构造出计算矩阵:

$$Y=\begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix},$$

其中,$b_{ij}=s_i-s_j,i,j=1,\ldots,n$.

通过 S 构造的 Y 记为 $Y=Con(S)$.至此,原问题转化为如下问题:

问题 1. 针对某个矩阵 X,如何确定向量 S,使其构造出的矩阵 Y 令 $f(X+Y)$最小?

由于该问题无法在多项式时间内验证最优解,而 DIS 系统又有很强的实时性要求.因此,我们针对不同的实时性要求提出两种回拨时钟值的计算方法,从而达到快速收敛到近似最优解的目的.

3.2 基于全局信息的快速拟合法

针对时钟回拨方法改善系统响应能力的最优系统状态,有如下定理:

定理 2. 针对问题 1,有

$$\forall S \ \text{ s.t. } f(X+Con(S))\geq \frac{Sum(X)}{n-1},$$

其中,$Sum(X)$代表矩阵 X 所有元素的和,即

$$Sum(X)=\sum_{i=1,\ldots,n,\ j=1,\ldots,n} a_{ij}\ .$$

证明:对于任意的 S,令 $Y=Con(S)$,对于矩阵 $X+Y$,我们取出每行元素的最大值组成集合 NS_1,再取出每行元素的次大值组成集合 NS_2,依此类推,共得到 n 个集合 $NS_1,NS_2,\ldots,NS_n$,可知 $Sum(NS_1)\geq Sum(NS_2)\geq\ldots\geq Sum(NS_n)$.由 f 函数定义可知 $f(X+Y)=Sum(NS_1)$,而由于 $X+Y$ 对角线元素为 0,可知 $Sum(NS_n)=0$.假设

$$f(X+Y)=Sum(NS_1)<\frac{Sum(X+Y)}{n-1},$$

则有

$$Sum(X+Y)=Sum(NS_1)+Sum(NS_2)+\ldots+Sum(NS_{n-1})<(n-1)\cdot\frac{Sum(X+Y)}{n-1},$$

上式中 $Sum(X+Y)<Sum(X+Y)$矛盾,因此假设不成立,可知下式成立

$$f(X+Y)\geq\frac{Sum(X+Y)}{n-1}.$$

又由于 $Sum(Y)=0$,故 $Sum(X+Y)=Sum(X)+Sum(Y)=Sum(X)$,可得

$$f(X+Y)\geq\frac{Sum(X)}{n-1}.$$ □

通过定理 2,我们得到了通过回拨时钟可以达到的系统的响应能力不会低于某一下限这一结论.并且,这种理论最优状态在大多数情况下是无法实现的,因为这要求原始矩阵 X 除对角线元素外各列对应元素差相等,即

$$\forall i_1,i_2,k,l(i_1,i_2,k,l\in[1..n])\wedge(i_1\neq k)\wedge(i_2\neq k)\wedge(i_1\neq l)\wedge(i_2\neq l)\Rightarrow a_{ki_1}-a_{ki_2}=a_{li_1}-a_{li_2}\ .$$

上述条件在实际系统中基本上无法得到满足,因此我们引入下述定理,并通过该定理设计我们的实时近似最优算法.

定理 3. 若对于矩阵 X 存在最优状态目标矩阵的构造向量 S',即

$$\exists S' \quad \text{s.t.} \quad \forall S\Rightarrow f(X+Con(S))\geq f(X+Con(S')).$$

令 $Y'=Con(S')$,则对于矩阵 $X+Y'$的元素 $c_{ij}(i,j=1,\ldots,n)$,有

$$\forall i_1,i_2,j(i_1,i_2,j\in[1,\ldots,n])\wedge(i_1\neq i_2)\wedge(\forall k(k\in[1,\ldots,n])\wedge(k\neq j)\Rightarrow(c_{i_1j}>c_{i_1k}))\Rightarrow(\exists l(l\in[1,\ldots,n])\wedge(l\neq j)\wedge(c_{i_2j}\leq c_{i_2l}))\ .$$

证明:若定理 3 不成立,则下式成立:

$$\exists i_1,i_2,j(i_1,i_2,j\in[1,\ldots,n])\wedge(i_1\neq i_2)\wedge(\forall k(k\in[1,\ldots,n])\wedge(k\neq j)\Rightarrow(c_{i_1j}>c_{i_1k}))\wedge(\forall l(l\in[1,\ldots,n])\wedge(l\neq j)\Rightarrow(c_{i_2j}>c_{i_2l}))\ .$$

假设 i_{1f},i_{2f},j_f 满足上式,则存在一个足够小的Δ,使得对应项减去Δ后上式仍然成立,即

$$\exists\Delta(\Delta>0)\wedge(\forall k(k\in[1,\ldots,n])\wedge(k\neq j)\Rightarrow(c_{i_1j}-\Delta>c_{i_1k}))\wedge(\forall l(l\in[1,\ldots,n])\wedge(l\neq j)\Rightarrow(c_{i_2j}-\Delta>c_{i_2l}))\ ,$$

则对于构造最优状态的向量 S',设 $S'=(s'_1,s'_2,...,s'_{j_f},...,s'_n)$,令 $S''=(s'_1,s'_2,...,s'_{j_f}+\Delta,...,s'_n)$,则其构造的矩阵 $X+Y''$相对于原来的构造矩阵 $X+Y'$在第 i_{1f}和 i_{2f}行上最大值均减小Δ,而在第 j_f行最大值增加Δ,最终得到

$$f(X+Con(S''))=f(X+Con(S'))-\Delta<f(X+Con(S').$$

与题设矛盾,因此假设不成立,定理 3 成立.证毕. □

定理 3 说明了系统最优状态的必要条件,因此我们可以利用该定理设计快速近似算法,下面首先给出基于全局延迟信息的迭代算法,其过程描述见算法 1.

算法 **1**. 基于全局信息快速迭代算法.

输入:节点延迟矩阵 X,各节点最大容忍延迟向量 CFM;

输出:回拨时钟值向量 S.

```
IterativeComputation(X,CFM,S)
1  repeat
2     IterativeS[1,…,n]={0}              //初始化数组
3     Y[1,…,n,1,…,n]={0}
4     IncreaseIter(IterativeS,X,CFM)
5     if IterativeS!={0}                 //根据迭代值构造新的延迟矩阵
6     {
7        for i=1 to n
8        for j=1 to n
9           Y[i,j]=IterativeS[i]–IterativeS[j]
10       X=X+Y
11       S=S+IterativeS
12    }
13 until IterativeS={0}
14 repeat
15    MinS[1,…,n]={BIGINT}
16    DecreaseIter(MinS,X)
17    S=S–MinS
18 until MinS[1,…,n]={0}
19 算法结束
```

算法 1 主要包括递增迭代和递减迭代两部分,其子程序描述如下:

```
IncreaseIter(IterativeS,X,CFM)
1     MaxS[1,…,n]={0}
2     for i=1 to n
3     {
4        寻找矩阵 X 第 i 行的最大值 MAXi
5        记录该最大值的编号 ni 以及与次大值之间的差值 di
6        if di>MaxS[ni]
7        {
8           IterativeS[ni]=MaxS[ni]
9           MaxS[ni]=di
10       }
11       else if di>IterativeS[ni]
```

```
12          IterativeS[ni]=di
13    }
14    for i=1 to n
15    {
16      if (MAXi+IterativeS[i])>=CFMi
17          IterativeS[ni]=max(CFMi−MAXi,0)
18    }
DecreaseIter(MinS,X)
1     for i=1 to n
2     {
3       寻找矩阵 X 第 i 行的最大值 MAXi
4       for j=1 to n
5         if (MAXi−X[i,j]<MinS[j])
6           MinS[j]=MAXi−X[i,j]
7     }
```

定理 **4**. 针对算法 1 的输入 X 和输出 S,有 $f(X+Con(S))\leq f(X)$.

证明:分别讨论算法 1 中上、下两个循环迭代过程.

针对上面的循环迭代过程,每次迭代过程计算出 *IterativeS*,对其中任意的 *IterativeS*[i]>0,等价于对原始矩阵的第 i 行全部增加 *IterativeS*[i],而在第 i 列都对应减去 *IterativeS*[i].由算法 1 可知,只有在所有行的最大值有 2 个以上在第 i 列时,*IterativeS*[i]取值才能大于 0,因此对矩阵 X 的操作将导致第 i 行最大值增加 *IterativeS*[i],而其他至少有 2 行的最大值减小 *IterativeS*[i],因此得到新的 f 值函数至少减小了 *IterativeS*[i].因此可知,上面迭代出的 S 满足 $f(X+Con(S))\leq f(X)$.

针对下面的循环迭代过程,每次迭代过程计算出 MinS,对其中任意的 MinS[i]>0,等价于对原始矩阵的第 i 行全部减去 MinS[i],而在第 i 列都对应增加 MinS[i].由算法 1 可知,只有第 i 列在所有行均不是最大值的时候,MinS[i]取值才能大于 0,因此对矩阵 X 的操作将导致第 i 行最大值减小 MinS[i],而其他行的最大值保持不变,因此得到新的 f 值函数至少减小了 MinS[i].因此可知,下面迭代出的 S 满足 $f(X+Con(S))\leq f(X)$. □

定理 4 证明了算法 1 在保持 DIS 系统功能的基础上,能够有效改善系统的响应能力.然而,该算法需要知道各节点间消息通信延迟的全局信息,而且要集中大量数据统一进行计算,不利于分布式系统的部署和实现.因此,我们设计了基于分布条件下的快速近似算法.

3.3 基于局部信息的逐步逼近法

在分布环境下,各节点只能获知本节点与其他节点之间通信延迟情况,而无法获知整体各个节点间的通信延迟信息.因此,我们针对单个节点的局部延迟信息,改进原来的集中式算法为分布式算法.该算法分为初始化过程、探测过程和计算过程 3 个部分.

算法 **2**. 基于局部信息的逐步逼近算法.

```
InitProcess()                    //初始化过程
{
    初始化数组 S[1,…,n]={0}
    计时器清零
}
DetectProcess()                  //探测过程
{
    if 计时器到达探测设定周期
```

```
    {
        探测自己的局部信息延迟,得到 X[i,1,…,n]
        寻找到 X[i,1,…,n]中的最大值 X[i,Maxi],并得到其与次大值的差 Di
        调用 SendMaxDelay(Maxi,Di),将延迟差 Di 发送至节点 Maxi
        计时器清零
    }
    Else
        计时器递增
}
ComputeProcess()            //计算过程
{
    收集各节点传送来的延迟差{Di}
    计算延迟差集合中的次大值 SecMaxDi
    探测自己的局部信息延迟,得到 X[i,1,…,n]
    寻找到 X[i,1,…,n]中的最大值 X[i,Maxi]
    If (X[i,Maxi]+S[i]+SecMaxDi)<=CFMi
        调整本地回拨时钟值 S[i]=S[i]+SecMaxDi
    Else
        调整本地回拨时钟值 S[i]=CFMi–X[i,Maxi]
    广播新的回拨时钟值 S[i]
}
```

算法 2 首先设置各个节点的回拨时钟值为 0,然后周期性地检测本节点的局部延迟信息,并将延迟最大者的编号和其与次大者之差发送出去,促使对应节点调整其回拨时钟值.由于算法 2 定期对网络延迟信息进行探测,因此可以有效地满足不断变化的网络条件,其多次执行结果相当于对算法 1 的逐步逼近,因此其有效性证明也可以从算法 1 的证明中得到.

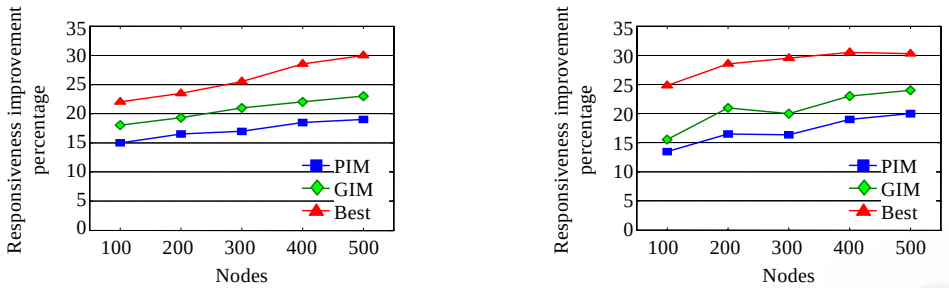
4 实验结果与分析

4.1 静态性能分析

我们首先验证算法的静态优化性能,通过 GT-ITM 网络拓扑生成工具,分别生成 Waxman 平面随机图模型和 Transit-Stub 层次模型拓扑来进行模拟验证.对于 Waxman 模型,采用参数α=0.6,β=0.4 的 Waxman 模型 1,节点数设置从 100 变化到 500.对于层次模型,设置 transit 域之间的边延迟为 30ms~50ms 均匀分布,transit-stub 域间边延迟为 50ms~100ms 均匀分布,stub-stub 域间没有边连接,stub 域内部节点间延迟在 20ms~40ms 均匀分布,节点数从 100 变化到 500.

在实验中,由于算法对不同延迟矩阵的优化程度不同,因此针对每种情况实验 20 次,记录每次生成的延迟矩阵优化结果并计算其平均值,其结果如图 4 所示.

图 4 中,Best 是针对生成的延迟矩阵进行覆盖性搜索计算出回拨时钟值,进而达到系统最优状态时相对原始矩阵的响应时间改进比例,GIM 是算法 1 针对延迟矩阵全局信息进行优化所得到的改进比例,而 PIM 是算法 2 根据局部延迟信息经过多次迭代稳定后所得到的改进比例.由图中曲线可以看出,回拨时钟方法对原有矩阵响应能力的改善平均可以达到 20%左右,而且其快速算法和全局最优的计算方法效果相差不大.



(a) Waxman model
(a) Waxman 模型

(b) Hierarchical model
(b) 层次模型

Fig.4 Responsiveness improvement

图 4 响应能力改进

4.2 动态性能分析

为验证算法在实际系统中的动态性能,我们在由 12 个节点组成的局域网中构建了 DIS 一致性模拟验证环境,底层以北京航空航天大学 BH-RTI[10,11]作为运行支撑平台,上层设计了以模拟大规模空战演练为目标的仿真应用,运行界面如图 5 所示.在仿真过程中,将全部的事件消息序类型设置为 RO 类型,并打开异步传输机制,由各节点对消息缓冲队列进行维护,从而尽可能地减小消息排队延迟.同时,通过中间层的延迟模拟器来模拟广域网节点间的通信延迟,其取值均值事先由拓扑生成器[12]加以构造,其延迟的时间分布特性满足负指数分布[13].

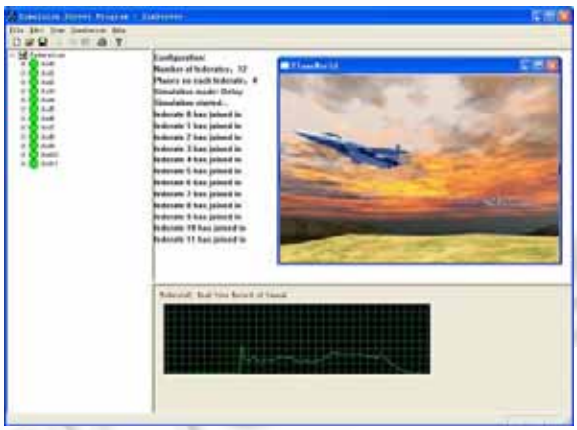
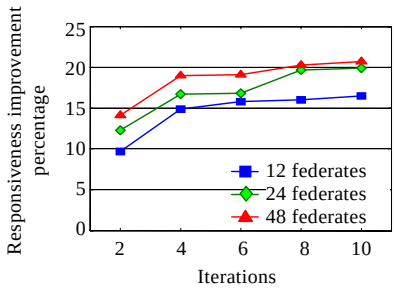


Fig.5 A screen shot of air war modeling experimental environment

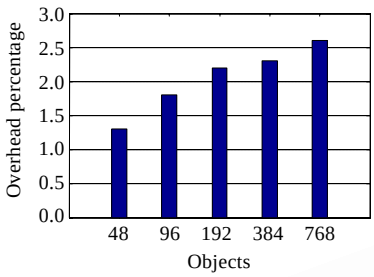
图 5 空战演练系统模拟实验环境运行界面

在实验过程中,由于回拨时钟会在交互发出时计算特定的执行时间戳,因此会增加一定的计算量.因此,我们在实验中一方面统计各节点响应能力的变化情况,另一方面对比是否采用回拨时钟方法在消息处理时间和发送延迟方面的差异,如图 6 所示.

由图 6 可知,算法 2 在迭代 10 次左右就能达到较好的优化效果,性能改进比例仍然可以达到 20%,而且其对原始消息处理和发送产生的额外开销可以保持在 3%以内,基本上没有造成较大的影响.



(a) Iteration method
(a) 迭代方法



(b) Overhead of call-back clock method
(b) 回拨时钟方法的开销

Fig.6
图 6

5 结束语

分布式交互仿真通过支持多用户的实时交互来提供一种可沉浸的虚拟空间,而分布式交互仿真中的一致性问题则严重影响着系统的功能和性能.现有的面向连续模型的一致性控制方法是通过降低节点的响应能力来达到一致性的功能需求的,在规模增大时会严重影响系统的性能.本文提出基于回拨时钟的异步时钟一致性控制方法,在不影响系统功能的前提下改善了系统的性能,提高了分布式交互仿真的适用性.

本文提出的回拨时钟方法在整体提高系统响应能力的同时,会增加部分节点的响应时间,这需要在回拨时钟值计算过程中加以限制,或者针对不同节点上实体的不同响应需求来分别加以计算.另外,在大规模网络中实时地进行准确的延迟探测和统计也存在着一定的困难,需要在进一步的工作中加以解决.

References:

[1] Deb F. Distributed interactive simulation: It's past, present, and future. In: Charnes JM, Morrice DJ, Brunner DT, Swain JJ, eds. Proc. of the 1996 Winter Simulation Conf. Washington: IEEE Computer Society, 1996. 179−185.

[2] Declan D, Tomas W, Seamus M. On consistency and network latency in distributed interactive applications: A survey—Part I. Presence: Teleoperators and Virtual Environments, 2006,15(2):218−234. [doi: 10.1162/pres.2006.15.2.218]

[3] Fujimoto RM. Parallel and Distributed Simulation Systems. Wiley Interscience, 1999.

[4] Zhou SP, Cai WT, Stephen JT, Bu-Sung L, Wei JH. Critical causal order of events in distributed virtual environments. ACM Trans. on Multimedia Computing, Communications and Applications, 2007,3(3):151−169.

[5] Cai WT, Stephen JT, Bu-Sung L, Zhou JL. An alternative time management mechanism for distributed simulations. ACM Trans. on Modeling and Computer Simulation, 2005,15(2):109−137. [doi: 10.1145/1060576.1060577]

[6] Saul PH, Jean F, Khalil D. The inmediate dependency relation: An optimal way to ensure causal group communication. Annual Review of Scalable Computing, 2004,6(3):61−79.

[7] Thomas AF. RING: A client-server system for multiuser virtual environments. In: Zyda M, ed. Proc. of the ACM Symp. on Interactive 3D Graphics (SIGGRAPH). New York: ACM, 1995. 85−92.

[8] Qin X. Delayed consistency model for distributed interactive systems with realtime continuous media. Journal of Software, 2002, 13(6):1029−1039. http://www.jos.org.cn/1000-9825/13/1029.htm

[9] Mauve M, Vogel J, Hilt V, Effelsberg W. Local-Lag and timewarp: Providing consistency for replicated continuous applications. IEEE Trans. on Multimedia, 2004,6(1):47−57. [doi: 10.1109/TMM.2003.819751]

[10] Zhang Y, Zhou Z, Wu W. A hierarchical time management mechanism for HLA-based distributed virtual environment. Journal of Computational Information Systems, 2006,1(2):7−15.

[11] Zhao QP, Zhou Z, Lü F. Algorithm of simulation time synchronization over large-scale nodes. Science in China (Series F): Information Sciences, 2008,51(9):1239−1255.

[12] Cheng L, Hutchinson NC, Ito MR. RealNet: A topology generator based on real internet topology. In: Proc. of the 22nd Int'l Conf. on Advanced Information Networking and Applications Workshops. 2008. 526–532. http://portal.acm.org/citation.cfm?id=1395080.1395414

[13] Kata J, Shimizu A, Goto S. Active measurement and analysis of delay time in the Internet. In: Proc. of the 1999 Int'l Workshops on Parallel Processing. 1999. 21–24. http://portal.acm.org/citation.cfm?id=848220

张伟(1982－),男,吉林榆树人,博士生,主要研究领域为分布式虚拟环境,分布式电路仿真.

彭宇行(1963－),男,博士,教授,博士生导师,主要研究领域为并行与分布式处理.

周航军(1979－),男,博士生,主要研究领域为分布式虚拟环境.

李思昆(1941－),男,教授,博士生导师,CCF 高级会员,主要研究领域为虚拟现实与可视化.