文章编号:1001 - 9081(2011)05 - 01413 - 04
doi:10.3724/SP.J.1087.2011.01413

基于远程过程调用和壁垒同步的分布式离散事件仿真模型

陈优子,陈俊延,王　彤
(北京航空航天大学 电子信息工程学院,北京 100191)
(yoyocyz@126.com)

摘　要:针对多处理器高数据量情况的并行离散事件仿真提出了一种高效的仿真模型。仿真采用时间步进的推进方式,运用远程过程调用(RPC)分布调用机制实现仿真成员之间的互操作。仿真中采用壁垒同步机制保证仿真推进时的时间同步,以确保因果关系正确。实践证明,在保证了仿真准确性的前提下,该模型可对传输层和网络层协议进行仿真测试,能够对大数据量进行准确实时的处理,并在一定程度上提高了仿真效率。

关键词:并行离散事件仿真; 远程过程调用机制; 保守策略; 时间单位推进; 壁垒同步
中图分类号: TP391.9　　**文献标志码**:A

Distributed discrete event simulation model based on RPC and barrier synchronization mechanism

CHEN You-zi, CHEN Jun-yan, WANG Tong
(*School of Electronic and Information Engineering, Beihang University, Beijing* 100191, *China*)

Abstract: A distributed simulation approach was proposed for discrete-events simulation with considerable amounts of events between logical processes. The proposed approach employed a time-driven method to simulate occurrence of discrete-events, using Remote Procedure Call (RPC) to describe the interaction between simulation members. In this approach, barrier synchronization objects were deployed for time synchronization in simulation advancement, in order to ensure the correctness of the causal ordering. Results obtained from the experiments show that the proposed approach can correctly and promptly handle large number of events, providing accuracy guarantee and efficiency improvement of the simulation model.

Key words: Parallel Discrete Event Simulation (PDES); Remote Procedure Call (RPC); conservative strategy; time-unit advance; barrier synchronization

0　引言

并行离散事件仿真(Parallel Discrete Event Simulation, PDES)是指在多处理器系统上并行执行离散事件的仿真模型,通过将负载分发到多处理器上处理提高仿真效率,同时运用一定的时间同步策略调度各节点上的并发事件,以保证仿真按照正确因果顺序顺利进行[1]。并行离散事件仿真中,系统的状态只在事件的发生时产生变化,而不是随时间不断地改变系统状态,因而在分析仿真方面比以高层体系结构(High Level Architecture, HLA)为代表的分布式交互仿真具有相对较高的运行效率[2]。如果离散事件发生频率很高,系统数据量较大时,传统的仿真机制会有很大的负载,运行效率和准确性也会相对降低。本文针对多处理器高数据量情况的离散事件仿真,提出了一种可以较高效率执行的仿真模型,运用远程过程调用(Remote Procedure Call, RPC)分布调用机制实现仿真成员的交互以及从服务端取得步进值,采用壁垒机制实现成员间同步,较好地提高了仿真的高效性准确性。

1　系统设计原理

1.1　RPC 调用机制

RPC 是一项广泛用于支持分布式应用程序(不同组件分布在不同计算机上的应用程序)的技术。RPC 的主要目的是为组件提供一种相互通信的方式,使这些组件之间能够相互发出请求并传递这些请求的结果[3]。

RPC 是一种会话层协议,通过网络从远程计算机程序上请求服务,而不需要了解底层网络技术的协议。它在两个网络进程之间建立一条逻辑信道进行会话连接,并利用这个信道进行信息交换。在 OSI 网络通信模型中,RPC 跨越了传输层和应用层,提高了程序的互操作性,使得开发包括网络分布式多程序在内的应用程序更加容易,所以本模型使用 RPC 分布式调用机制实现仿真推进。

1.2　时间管理和推进

根据仿真要求,为了保证仿真过程中事件处理顺序不偏离物理系统中事件发生的顺序,以及减轻开发者负担,我们采用保守的时间推进机制。

保守时间推进机制中,规定了时间前瞻量(Look Ahead)和时戳下限值(Lower Bound Time Stamp, LBTS)。该算法的基本思想是,设定前瞻量 L, 各逻辑过程(Logic Process, LP)通过前瞻量计算出将来可能收到消息的时戳下限值 $LBTS$, 假设当前局部仿真时钟最小的 LP 的仿真时钟为 T_{min}, 则时戳在区间 $[T_{min}, LBTS]$ 的事件都可以被安全执行[4]。

离散事件系统存在两种基本推进方式[5]:以时间为单位

收稿日期:2010 - 10 - 26;**修回日期**:2011 - 02 - 02。

作者简介:陈优子(1986 -),女,河北衡水人,硕士研究生,主要研究方向:无线网络、航空电子网络; 陈俊延(1983 -),男,河南南阳人,博士研究生,主要研究方向:无线网络、航空电子综合; 王彤(1962 -),女,山西太原人,研究员,主要研究方向:航空电子综合系统、总线通信网络、电子设备智能测试。

推进(Time-Unit Advance)和以事件为单位推进(Event-Unit Advance)。对于事件发生时间间隔较短且在时间轴上分布较均匀时,时间为单位推进方式可以在保证精度的同时获得较高的效率[6]。本模型设计对通信协议或者路由协议进行仿真,在较小的发送周期内信息数量相对较大,所以采用时间为单位推进(时间步进)的方法,即选择适当的时间单位(必须足够小)作为固定的时间推进步长。时间步进的原理是此次步进所要处理的所有事件都认为在一个事件点发生,新产生的所有事件都认为在下一次步进发生,所以可以将推进步长值作为时间前瞻量。每推进一步都作如下处理:首先检查事件列表内有无事件在该时间推进步长内发生,若没有,则仿真时钟向前推进一个时间单位;若有,则处理事件,相应地改变系统状态,然后向前推进一个时间单位。如果该时间步进内有多个事件发生,用户必须事先规定好处理各类事件的优先级。

1.3 信息传递机制

在多用户分布仿真中,需要通过网络传递仿真信息,包括状态信息和命令信息等。为了使信息可靠地发送到目的地,以便有效地操作、控制相应的机器,本文采用了“信息报文”方式传送信息。报文采用统一的格式,包括时戳、信息类型、节点 ID、发送地址、目的地址、数据信息等内容,每个部分占用字节数都统一规定。

1.4 壁垒同步

壁垒是在多线程应用程序中实现线程同步的一种机制,由一个计数器、一个信号列表和至少一个互斥锁组成。多线程程序中各线程执行不同的任务,线程之间是相对独立的,但是有时一个任务需要某些线程同步执行,这时就要在多线程运行过程中建立一个壁垒。对于每个到达此壁垒的线程,如果不是最后一个到达的线程,会停在壁垒前等待信号;如果是最后一个到达的线程,则触发信号打开壁垒使所有线程通过。

2 仿真平台架构

2.1 仿真平台结构

并行离散事件仿真系统模型如图 1 所示。

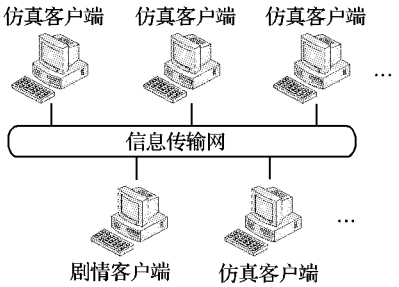

图 1 并行离散事件仿真系统模型示意图

1)剧情服务器。设置仿真系统的仿真初始状态,包括各仿真成员属性信息(位置、速度、方向、设备状态等)和网络拓扑状态;控制仿真的开始和结束,仿真开始后作为服务器响应各节点的请求,各节点都请求后下发时钟。

2)仿真客户端。仿真成员地位平等,仿真开始后,各仿真成员分别执行时间步进内的事件,事件执行结束后向仿真服务器发送消息请求推进,并等待服务器响应。从仿真服务器取得步进时钟后开始下一步仿真,仿真继续进行。

2.2 仿真流程设计

根据仿真时间一致性要求,设计仿真流程如图 2 所示。

1)初始化时,各客户端第一次申请推进时,其客户信息被存到服务端的成员列表中,客户端信息包括客户端的平台号和一个表示客户端是否已请求推进的状态标志(即请求状态,已经请求推进时此状态置位,请求结束后此状态复位)。

2)客户端调用服务器的等待推进函数 WaitForCallRequest(),WaitForCallRequest ()函数为异步远程过程调用函数,其参数中有一个输出参数为此次下发的时钟值,同时此客户端对应请求状态置位。

3)所有客户端都请求推进后,异步推进函数返回,各客户端取得此次仿真时钟,开始下一次仿真步进,同时客户端退出本次取时钟队列,其对应请求状态复位。

4)本次请求结束,客户端执行完本次时钟步进内的所有事件后再次请求推进,调用 WaitForCallRequest() 函数并且请求状态再次置位,依次进行直到事件全部结束仿真推进结束。

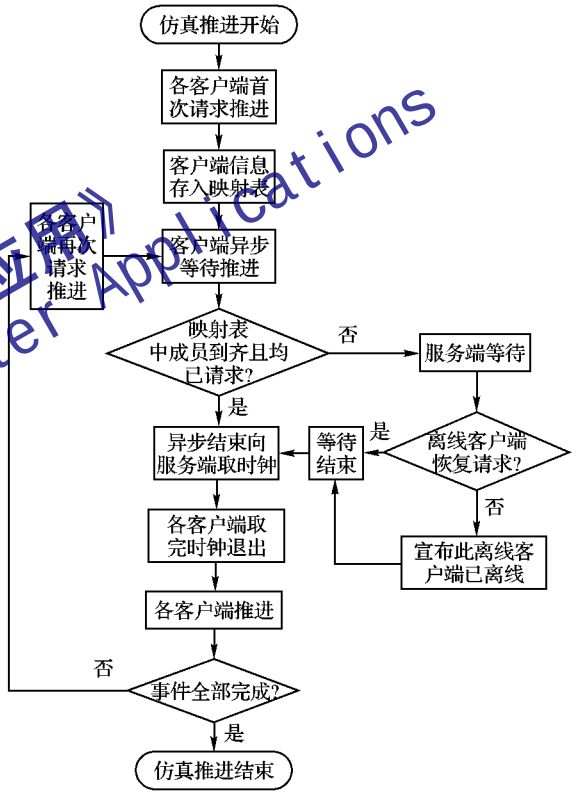

图 2 仿真流程设计

2.3 仿真模型实现

根据仿真设计要求,各仿真成员在处理完成推进时间步长范围内的事件后需要向服务器请求再次推进,服务器等待所有成员都发起请求后下发时钟。客户端状态机见图 3。

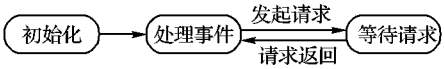

图 3 客户端状态机

仿真服务器需要建立一个壁垒(Barrier)来等待所有仿真成员都请求推进之后才可以下发时钟,而且需要保证所有客户端在某个时刻之前都已突破此壁垒,才能正常进入下一轮推进。但是,只有一个壁垒在实际操作上很难做到,因为壁垒被突破后并重建时,如果仍有成员未通过,则这些成员就参与到下一轮推进,造成推进不同步的情况;更甚者,容易掌握不好壁垒重置时机,导致死锁。如果不能保证同时突破,就要建

立第二个壁垒控制重置时机。按照实际的可操作要求,对成员推进机制进行详细设计,改进为两个壁垒后的客户端状态图如图 4。

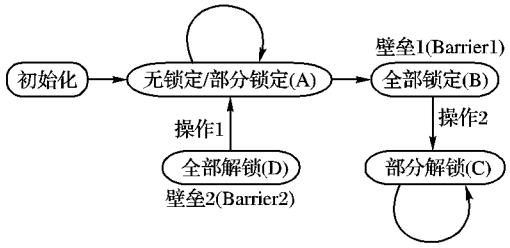

图 4　客户端改进状态图

图 4 中:

操作 1　欲推进成员列表重置,加互斥锁。

操作 2　已推进成员列表重置,解互斥锁。

从状态图上可以看出,成员推进过程中等待时钟下发的时候设置了壁垒 Barrier1,而且在状态 C→状态 D 过程中设置了另一个壁垒 Barrier2。两个壁垒起到互相保护的作用,确保所有成员均已突破上一壁垒时,才允许某个成员开始突破下一壁垒:

Barrier1:确保任意一个成员将要突破自己时,其他所有成员至少已突破上一轮的 Barrier2。

Barrier2:确保任意一个成员将要突破自己时,其他所有成员至少已突破 Barrier1。

Barrier2 的作用是确保所有成员都能突破上一个壁垒,同时进入下一轮推进。如果 C→D 过程中不设壁垒,则在有些成员未解锁之前已解锁的其他成员进行下一次推进请求时就会造成死锁。

2.3.1　壁垒机制实现

状态 A→状态 B 的壁垒 1(Barrier1)使用互斥锁来实现。互斥锁由一个事件和等待该事件的 API——WaitForSingleObject 组成。WaitForSingleObject()会挂起调用线程直至内核对象变成有信号态。由于所有成员均请求推进后才能完成 A→B 的状态转化,成员请求推进的时刻不定,加锁的时间可能会很长,这种情况下使用互斥锁比较节省资源。WaitForSingleObject()等待的事件是人工重置事件,通过 SetEvent()解锁,Barrier1 解除;并在特定时刻通过 ResetEvent()加锁,Barrier1 建立。

状态 C→状态 D 的壁垒 2(Barrier2)使用循环锁来实现。循环锁通过一个 while 循环不断检查 bool 类型的状态标记实现。C→D 的状态转化条件是满足所有成员突破“Barrier1”(“Barrier2”用于检测这一条件),等待的时间较短。循环锁占用资源远大于互斥锁,但此过程中加锁时间极短可以忽略资源占用的弊端。使用循环锁的目的是对状态改变尽快做出反应。

2.3.2　状态检测器实现

为了确实每个状态是否准确到达,在状态 A→状态 B 和状态 C→状态 D 之间设置了两个检测器,即两个映射表。

1)欲推进成员列表。用于确定状态 B,这个映射表中保存了当前有推进请求并在 Barrier1 前等待下发步进时钟的成员。如果列表为满,说明所有成员都已经请求了推进,到达 B 状态可以拆除 Barrier1,即使用 SetEvent()函数设置事件为有信号状态。

2)已推进成员列表。用于确定状态 D,这个映射表中保存了已经请求推进并且获得步进时钟的成员,即刚刚突破 Barrier1 的成员。如果列表为满,说明所有成员已经突破 Barrier1,已经对循环锁解锁(即拆除 Barrier2)到达 D 状态。D 状态之后需使用 ResetEvent()将事件设置为无信号状态对互斥锁加锁建立起 Barrier1,以确保在下一轮推进正常进行。

状态 B→状态 C 和状态 D→状态 A 没有检测器,是无条件转换。

2.3.3　状态检测器重置时机

由 2.3.2 节可看出,两个映射表是随着推进的进行而不断变化的,映射表为满时突破壁垒,映射表是否为空也是壁垒重建的重要依据,所以它们的重置时机是保证推进顺利进行的重要因素。

理论上,欲推进成员列表重置可以在状态 B→状态 C→状态 D→状态 A 之间的任一环节,最晚要在“Barrier1”的建立之前;已推进成员列表重置可以在状态 D→状态 A→状态 B→状态 C 期间,且最晚要在“Barrier1”的解除之前。

为了状态机的可靠性和操作的可行性,最好是保证只有一个成员有权利拆除壁垒,保证所有成员都通过壁垒后才可以重建。由于两个壁垒既相互协助又相互制约,如果映射表重置时机过早,例如在壁垒突破但有些成员尚未通过壁垒时,有可能造成死锁的情况。

一个保险的方法是用一个临时变量保持检测器的状态,保证所有成员安全通过之前此临时变量状态不变。欲推进成员列表的临时变量最好是在状态 D→状态 A 之间重置,最晚要在“Barrier1”的建立之前;已推进成员列表的临时变量最好是在状态 B→状态 C 期间重置,且最晚要在“Barrier1”解除之前。

详细仿真流程如图 5 所示。

3　仿真性能分析

实验仿真场景是由 25 个节点组成的示例网络,对无线自组网进行仿真。各节点推进过程中响应各种事件,包括人机交互事件、节点属性事件、报文事件等,其中报文事件是主要的事件来源,各节点利用大量的无线通信报文进行通信,包括各节点产生和转发的报文,周期性轮询报文和其他一些随机产生的报文事件。

实验测试对象为本模型时间步进机制前瞻量 50 ms 实例,因为本模型采用的保守策略,所以对比测试对象为保守策略下事件步进机制前瞻量 5 ms 实例与事件步进机制前瞻量 50 ms 实例。图 6 是在不同报文平均间隔下,3 种测试机制仿真运行时间比较图。由图 6 可以看出时间步进机制在报文平均间隔小于 100 ms 时,仿真运行时间均比其他两种机制要短。事件步进前瞻量 5 ms 情况下,随着报文频率的增大,效率会越来越低。这是因为报文事件可能使其他节点产生新事件,故前瞻量的选择要以报文传输时延为基准,而如果以理想情况下 5 ms 的平均传输时延作为前瞻量,会造成并行性严重下降。随着报文频率的增加,交互性报文也增加,采用事件推进机制时系统的计算量也会大大增加,仿真效率也会变低[8],因此有必要在保证一定仿真精确性的同时,适当增加前瞻量的值。根据我们的实验场景,最终选取 50 ms 作为前瞻量。同时要注意盲目增加前瞻量可能会导致发生因果关系错误的概率大大增加。而时间步进一方面便于控制各节点仿真时钟保持同步,另一方面每次步进中的所有事件都在同一

仿真时间发生,能够有效避免因果关系错误。故前瞻量 50ms 时选择时间步进作为推进机制更为合适。实验结果证明,在报文周期很短的情况下,时间步进的仿真执行效率要比事件步进高,且能够避免前瞻量设置过大而出现因果关系错误。

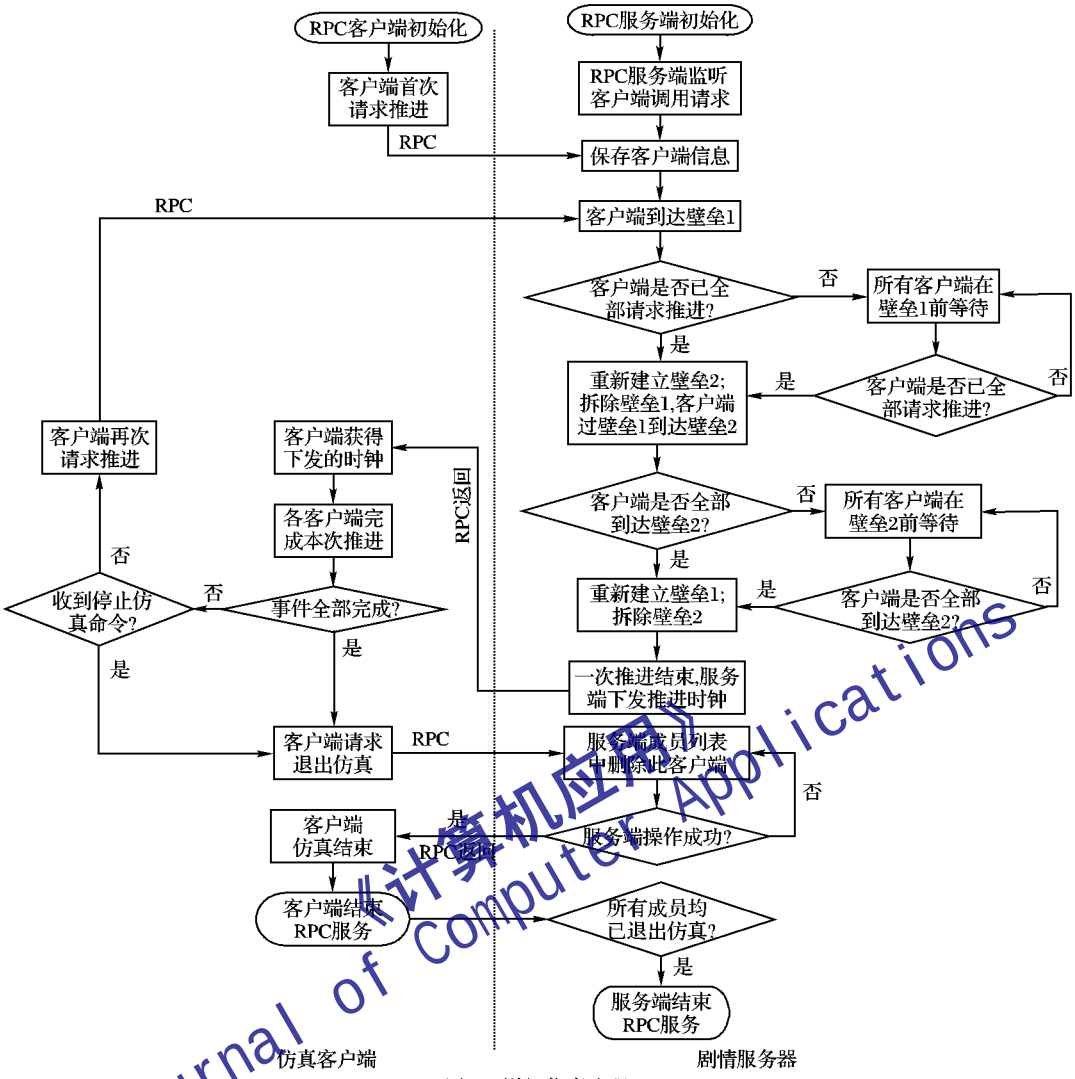

图 5 详细仿真流程

4 结语

本文介绍的利用 RPC 机制并基于保守时间策略的分布式离散事件仿真模型,在避免死锁和保证事件因果顺序的前提下,实现了多处理器高数据量分布式仿真系统,并且已配合仿真图形化界面,配合上层路由和传输协议在某高数据量分布式仿真系统中得到了应用。实践表明,此模型对大数据量进行准确实时的处理,有效地提高了并行仿真系统的可操作性和可实现性,具有较高的集成仿真效率。

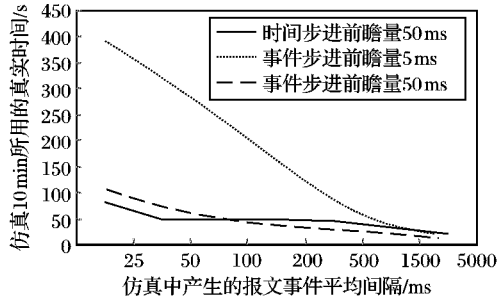

图 6 仿真效率随发包频率变化的曲线

参考文献:

[1] FUJIMOTO R M. Parallel discrete event simulation [J]. Communications of the ACM, 1990, 33(10): 30 - 53.

[2] FUJIMOTO R M. Parallel and distributed simulation systems [M]. New York: John Wiley and Sons, 2000.

[3] RFC1057, Remote procedure call, Version 2 [EB/OL]. [2010 - 09 - 01]. http://tools.ietf.org/html/rfc1057.

[4] 张文荣. 并行离散事件仿真对象测试技术的研究与实现[D]. 长沙: 国防科学技术大学, 2008.

[5] 黄柯棣, 张金槐, 李剑川, 等. 系统仿真技术[M]. 长沙: 国防科技大学出版社, 1998: 20 - 47.

[6] 黄洪. 一种新的离散事件仿真时间递进机制——混合时间递进机制[J]. 系统工程理论与实践, 1994, 14(4): 56 - 60.

[7] MAREJKA R. Barrier synchronization object for multi-threaded applications: United States, 7512950[P]. 2009 - 03 - 31.

[8] 马宝林, 孙济洲, 于策. 基于混合时间-事件驱动的信任值更新机制[J]. 计算机应用, 2006, 26(10): 2289 - 2290.

[9] 薛芳侠, 闫了了, 谢虹, 等. 分布式实时仿真系统高精度时间同步技术研究[J]. 计算机应用, 2006, 26(4): 989 - 994.

[10] PERUMALLA K S. Parallel and distributed simulation: Traditional techniques and recent advances [C]// Proceedings of the 38th Conference on Winter Simulation. Washington, DC: IEEE Computer Society, 2006: 84 - 95.