基于 JVM 的分布计算在网络仿真中的应用研究

王文鼐, 赵生妹
(南京邮电学院, 江苏南京 210003)

摘 要: 分析分式网络仿真系统的需求, 给出以 Java 虚拟机为平台的分布计算系统的对象设计结构, 并简要介绍所开发的仿真器的功能. 信元中继网的初步仿真表明, 采用分布式仿真方法, 5 台计算机协同仿真的效率可以提高 70% 以上.

关键词: 分布式计算; 通信网络仿真; 仿真器开发
中图分类号: TP39, TN91 文献标识码: A 文章编号: 0372-2112 (2001) 06-0804-04

On Networks Simulating by JVM-Based Distributed Processing

WANG Wen-nai, ZHAO Sheng-mei
(*Nanjing Institute of Posts and Telecommunicaitons, Nanjing, Jiangsu 210003, China*)

Abstract: The requirements of JVM (Java Virtual Machine)-based distributed computation for networks simulating were discussed in this paper. The architecture adopting central controlling mode and the functions of the developed simulator were described also. It was shown by a simplified example of cell-relay network that simulations, executed on 5 computers, are speeded up over 70%.

Key words: distributed processing; communication networks simulation; simulator implementation

1 引言

通信网络的仿真(或模拟) 在网络规划和设计、网络运行性能的研究、网络优化、网络发展需求预测, 以及网络新技术的研究等方面具有相当重要的作用[1]. 离散事件仿真(DES) 方法被广泛应用于系统仿真以及网络仿真之中. 在通信网络仿真方面, 近年来已经开发出若干专用的仿真软件包. 对于规模较大的通信网络和设备, 比如以宽带业务为主的通信网、每秒数百万次的数据报文或分组转换的交换设备, 由于仿真的计算量较大, 传输仿真软件平台的仿真速率只能依赖于主机的计算性能提高. 分布或并行计算的方法, 由若干主机协同处理计算任务, 是提高仿真计算效能的一个重要发展方向. 研究和开发分布式的通信网络仿真软件, 具有相当大的研究和应用前景.

在分布计算中, 相互间存在因果关系的事件, 由于不同类型事件的处理时间不尽相同, 计算单元之间的通信具有突发性, 因而会引发事件间的因果关系破坏, 从而使计算结果可能出现差错. 一般采用仿真时间, 即按事件的因果关系而排列的先后次序, 来表示仿真事件、处理单元的仿真过程. 对仿真事件在仿真时间上的先后顺序进行管理, 即因果关系控制, 是实现分布式网络仿真的一个关键因素[2]. 在离散事件仿真方法中, 单处理机方式在维持事件的因果关系上是自然的、简单的. 在应用分布或并行离散事件仿真(PDES; Parallel Discrete Event Simulation) 时, 因果关系控制的方式主要有两类. 一类要求仿真任务在分配给处理单元之前, 严格按因果约束关系排列[3]. 如果某处理单元的仿真时间先于仿真任务的仿真时间, 则该处理单元要停止等待. 另一类对仿真任务不作严格的因果约束, 但要求处理单元在接收到新的仿真任务. 另一类对仿真任务不作严格的因果约束, 但要求处理单元在接收到新的仿真任务, 而该任务在因果关系上又先于处理过的任务(事件), 将仿真状态摇回(roll-backing), 收回违背因果关系的已仿真过的任务, 以恢复并维持事件间的正常因果关系[4].

在实用化的分布式仿真系统中, 特别是在基于因特网的分布式仿真中, 跨平台是一个基本要求, 仿真运算代码要求能不作修改地运行于不同的软硬件环境中. Java 语言和 Java 虚拟机(JVM Java Virtual Machine), 提供了实现异质环境分布计算的一相当有利的机遇[5]. 为研究和开发分布式仿真系统, 本文分析并设计分布式网络仿真系统的体系, 报道以 JVM 为平台而开发的软件包, 以一简化的信元中继网络为例说明该软件包的功能.

2 分布式仿真体系设计

JVM 是 Java 程序的运行平台. 同本地代码程序相比, Java 程序的计算效率虽然较低, 但在异质性分布计算环境中, Java 程序具有相当强的跨平台特性. 基于 Java 的分布处理方案, 主要有与公共对象请代理体系(CORBA) 相合的方式, 利用 JNI 的方式等. 对于网络仿真, 这两种体系在计算效率和针

收稿日期: 2000-01-13; 修回日期: 2000-07-11

对性方面的适用性并不太理想. 因此, 本体系的设计结合 TCP/IP 协议的软件接口, 在 JVM 平台之上直接构造, 如图 1 所示.

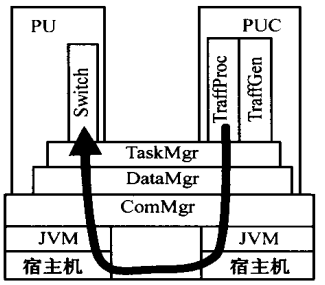

图 1　分布式网络仿真平台的体系示意图

分布式网络仿真平台, 在物理上由数目不限的处理单元(PU; Processing Unit) 和一个处理中心(PUC: PU Center) 组成. PUC 中, 类(class)"业务发生器" (TraffGen) 仿真产生网络的流入业务量, 并由类"业务处理" (TraffProc) 按仿真的时间先后进行排队. 类"TraffProc" 还负责按请求将仿真计算任务送到 PU 当中的类"交换机" (Switch), 而后者模拟交换功能, 将要返回的仿真任务交由 PUC 的类 TraffProc 作处理. TraffProc 的功能还包括因果约束判别和事件摇回控制; 仿真业务流在最终完成在目标网络中的转移之后, 从队列中移去; 以及仿真结果的统计和记录等.

仿真任务的产生由类"任务管理器" (TaskMgr) 根据业务队列的情况处理, 并负责将仿真任务送到相应的发出任务请求的 PU. 所有的分布计算信息都由类"通信管理器" (CommMgr) 最终完成在分布的处理机之间交换数据. CommMgr 以用户数据报协议(UDP) 为基础, 通过 Socket 实现异质宿主机环境下的通信.

由于 UDP 本身不具备可靠性, 也出于容错考虑, 在 TaskMgr 和 CommMgr 之间, 设计类"数据管理器" (DataMgr), 以处理在 PU 请求丢失时的故障, 以及因基础通信传送中断等而造成的死锁(deadlock).

Switch 仿真网络节点的交换功能, 对流入的以仿真任务表示的网络业务, 在完成交换过程的模拟后, 送回 PUC, 交由计算网络中其它 PU 作进一步交换或转换. 模拟交换时, Switch 根据目标网络的节点状态, 增加业务的延时, 或者当目标网络节点的资源占满时, 模拟业务丢失, 以仿真业务流的丢失率特性. 相应的, 目标网络节点的状态也因业务模拟而可能发生变化. 目标网络节点的状态信息以集中方式保存在 PUC. 所有 PU 从 PUC 得到目标网络节点的状态信息.

3　PDES 因果约束和控制方法

实现并行和分布式仿真, 需要制定仿真计算单元间交互的规则、划分并确定仿真模块(或对象) 的职能、以及系统内各部件之间的关系等. 交互规则包括, 计算单元的加入/撤离控制, 计算中间数据的存储控制, 计算单元间的通信协议, 容错处理和并发控制. 为提高分布仿真系统的可重用性, 需要建立良好的功能模块(或对象模块), 以及扩展仿真模块的接口. 在系统体系当中, 计算任务的同步控制(即因果约束) 是核心内容之一.

在传统 PDES, 主要按仿真事件在目标系统中的先后顺序, 判别因果关系. 我们认为, 经由不同网络路由的业务流, 在空间上并不相叠, 彼此之间不存在直接的因果关系, 可以由不同处理机并行仿真. 另外, 空间分布上存在重叠的、但不同时发生的多个业务流, 只要在网络状态上不相互依赖, 也可以被不同处理机并行仿真. 彼此不直接相关的仿真事件, 在仿真计算的过程中, 即使先生先后次序的颠倒, 从对计算结果进行统计的角度出发, 并不会违背因果约束. 以下以图 2 所示的网络为例, 作进一步说明.

图 2 中, 事件序列(e_1, e_3, e_4)和(e_2, e_5)表示两个不同的业务流. 如果以单处理机仿真, 则事件队列的变化可表示为:

$$(e_1, e_2)^{\rightarrow}(e_2, e_3)^{\rightarrow}(e_3, e_5)^{\rightarrow}(e_5)^{\rightarrow}()$$

在$(e_1, e_2)^{\rightarrow}(e_2, e_3)$的计算过程中, 事件 e_1 表示的业务转移由处理机仿真执行, 并产生 e_3, 以此类推, 直至事件队列为空, 即$()$. 从发生的时间(次序) 出发, 事件 e_5 必须在 e_3、e_4 处理完之后才能被仿真执行. 如上所述, 从网络仿真结果的真实性出发, e_5 在 e_3、e_4 前后被仿真计算都有是允许的. 换言之, 上述两个业务流之间, 不存在直接因果约束关系. 如果由两个处理机并行处理, 则总的事件队列的变化可以表示为:

$$(e_1, e_2)^{\rightarrow}(e_3, e_5)^{\rightarrow}(e_4)^{\rightarrow}()$$

显然, 通过分布处理方法, 可以相应地提高仿真效率.

当然, 如果存在另一个业务流, 在空间上与上述两个业务流交叉, 且发生因果约束, 则前述两个业务流可能存在间接因果约束关系. 对存在直接或间接因果约束的业务流仿真, 可以延用摇回(roll-backing) 方法恢复合理的时序[4].

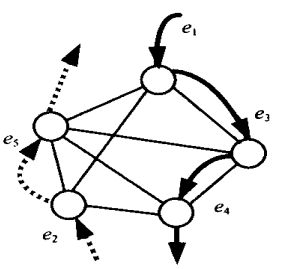

图 2　网络与业务流关系的示意图. 箭头线表示业务的流动路径, e_1 和 e_2 表示业务的流入事件, 下标表示业务流入时间上的先后次序.

因果关系的判别, 依赖于目标网络中的节点状态. 从仿真角度看, 目标网络节点状态, 可用排队模型当中的队列状态表示. 业务单元(比如单元) 流到目标网络中一个节点(即仿真事件), 如果不违背因果关系, 则经过一定延时后, 流向目标网络中的下一节点(即一个新的仿真事件). 此时, 上一节点的状态发生相应变化. 表示仿真事件先后和目标网络节点状态的参数, 是目标网络当中的时间, 为与仿真计算的实时间相区分, 一般称前者为仿真虚拟时间(virtual time)[4]. 虚拟时间控制是进行因果约束控制的关键. 虚拟时间的推进决定于仿真事件的虚拟时间, 即仿真业务流入到仿真网络节点的时间, 或者仿真业务流出的时间. 通过仿真事件虚拟时间与节点虚拟时间的比较, 可以较为容易地判别因果关系破坏, 通过摇回可进一步控制

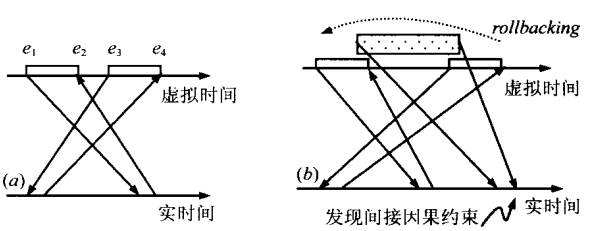

图 3　仿真任务时间顺序与实际仿真执行次序不一致的两种情况

以恢复正常的因果约束关系, 如图 3 所示.

在图 3(a) 的情况下, 如果事件 e_1 之前和 e_2 之后, 目标网络节点的状态没有变化, 即在仿真结果上不会影响已仿真计算过的事件 e_3, 或者说, e_3 与 e_1、e_2 并不存在直接因果约束, 则不需要作摇回处理. 这种在时间关系上存在先后, 但并不构成直接因果关系的事件, 不作摇回控制, 类似于隧穿(*tunneling*) 过程, 可称为虚拟时间隧道(*Virtual Time Tunneling*) . 而对于存在间接因果约束的仿真事件, 沿用通常的摇回机制还是可以恢复正常关系, 如图 3(b) 所示.

可以想见, 增加虚拟时间隧道机制, 可以减小摇回的频次, 提高仿真效率. 文献[6] 报道, 不作摇回处理, 可以提高总的分布仿真效率, 但会引发一定量的仿真错误. 本文引入的时间隧道机制, 不会产生文献[6] 的仿真计算错误.

4 分布仿真系统的功能

针对信元中继网所开发的分布式仿真系统, PUC 的用户界面部分如图 4 所示. 通过 PUC 的人机界面, 可以按要求建立相应的仿真目标网络的结构图. 所有 PU 通过注册而加入仿真计算. 在物理上分立的若干 PU 和 PUC 以 10Mb/ s 的计算机局域网互连. PUC 和 PU 均运行于 JVM 平台之上, 编译后的 Java 程序可以不作任何改动地在不同宿主机上运行(已测试通过的计算环境包括 Intel/ MS- WIN98, Intel/ LINUX, HP9000/ HPUX) .

PUC 的主窗口显示仿真目标网络的拓扑结构, 状态行显示仿真计算过程. 主窗口的菜单条包括仿真过程控制(中止、保存、上次中止状态参数的读取和继续等功能) , 网络配置控制(网络节点、网络链接的增减和修改等功能) , 仿真运行控制(业务流量设置、PU 加入允许控制等功能) , 以及一些提示信息等菜单项.

通过三个统计窗口, 可以实时观察仿真计算的中间结果. 监察图给出虚拟时间与实时间在计算过程当中的变化. 仿真吞吐量图, 以模拟完成的业务(信元) 数目与系统实际运行时间的变化关系, 反映系统的仿真计算效率. 业务统计图可以通过选择, 观察仿真目标网络中任意两端点的性能, 包括平均时延、时延变化和丢失率.

仿真目标网络的流入业务量以均匀分布的伪随机数产生算法为基础, 通过指数偏离计算, 得到符合泊松(Poisson) 分布的业务流. 业务强度可以在仿真计算过程中, 通过 PUC 实时改变, 以仿真观察业务强度变化与网络性能之间的动态关系.

本仿真平台具有一定的容错性能. 当某些 PU 与 PUC 出现物理通信短时中断, 或者特定仿真任务在计算超时从而使整个仿真计算停滞时, PUC 将主动恢复到一个可信的虚拟时间上, 以启动仿真计算继续进行.

5 信元中继网的仿真及分布计算效率结果

针对一个简单的信元中继网, 如图 5(a) 所示, 分布式仿真的计算效率可以通过仿真完成的业务数目(信元数) , 即模拟的吞吐量与实际运算的时间关系来反映, 如图 5(b) 所示.

图 5(a) 所示的目标网络中, 各节点均设为中央排队式交换机[7], 节点间的传输带宽均设为 155Mb/ s. 为观察分布式仿真平台的计算效率, 忽略传输时延对端到端时延及其对丢失率的影响, 以一个信元传输时间为单位的网络总流入业务量强度为 1. 2. PUC 运行的宿主机为 Intel PII 350 MHz/ MS- Win98 的 PC, 二个 PU 运行于同类宿主机; 另二个 PU 运行于主频为 130MHz 的 HP 工作站.

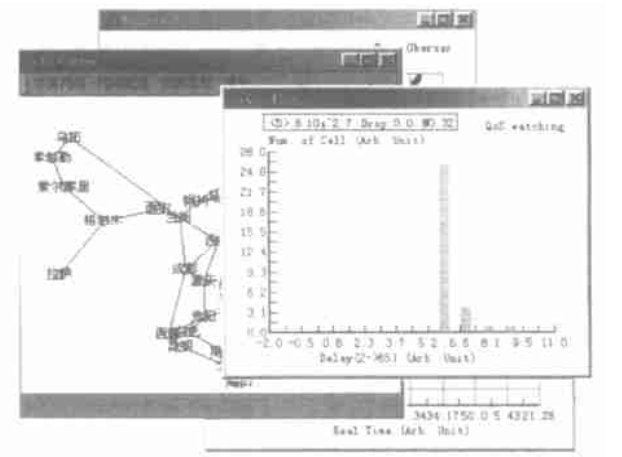

图 4 分布式网络仿真图 PUC 的用户界面

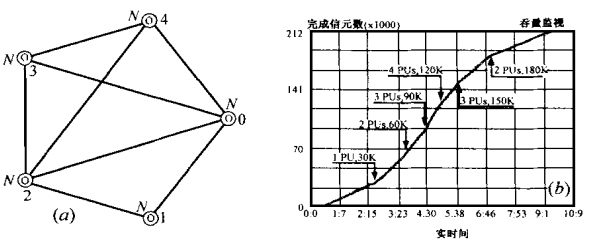

图 5 (a) 测试分布式仿真计算效率的例子网络; (b) 仿真完成的信元数与计算单元数的变化关系图.

图 5(b) 所显示的, 是完成 30000 个信元仿真的实际计算时间与参与仿真的 PU 数目关系. 可以看出, 按上述体系设计并开发的分布式仿真计算平台, 可以显著地提高仿真效率. 仿真计算时间与参与仿真主机数目之间的关系如图 6 所示.

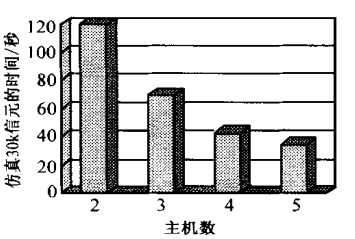

图 6 仿真计算时间与主机数的变化关系图

6 结论

本文简要分析了分布式网络仿真平台开发的基本需求. 为提高分布处理的计算效率, 根据网络运行特点, 在并行离散事件仿真方法中引入虚拟时间隧道机制. 本文给出一个中央控制的分布式网络仿真系统的设计方案, 并简要介绍了在此基础上开发的, 主要针对信元中继网、基于 JVM 平台中的分布式仿真系统的功能. 初步的测试表明, 该系统在加速计算效率方面有较好的性能, 5

台异质性主机协同计算的时间比单机的计算时间, 可以缩减 70% 以上.

由于 JVM 对 Java 语言程序是以解释运行方式执行, 因此, 在此基础上开发的分布式仿真系统同用本地(native) 代码编写的仿真系统相比, 在计算效率率上还存在一定差距. 随着 JVM 性能的提高, 以及适用于 Java 的硬件平台的开发, 基于 Java 的分布式网络仿真将具有相当广泛的应用前景.

参考文献:

[1] Law A M, and McComs M G. Simulation software for communications networks: The stae of the art [J]. IEEE Communications Magazine, 1994, 32(3): 44- 50.

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

[3] Chandy K M, and Misra J. Distributed simulation: A case study in design and verification of distributed programs [J]. Transaction on Software Engineering, 1979, 5(5): 440- 452.

[4] Jefferson D R. Virtual Time [J]. ACM Transaction on programming language and System, 1985, 7(3): 405- 425.

[5] Kreutzer W, Hopkins J and Mierlo M. SimJava: A framework for modeling queuing networks in Java [A]. Proc. of the 1997 Winter Simulation Conference [C], 1997: 483- 485.

[6] Rao D M, et al. Unsynchronized parallel discrete event simulation [A]. Proc. of the 1998 Winter Simulation Conference [C], 1998: 1563- 1570.

[7] Prycker M. 程时端, 刘斌译. 异步传递方式(第四章三节) [M]. 北京: 人民邮电出版社, 1995.

作者简介:

王文鼐　男. 1966 年出生于江苏南京. 于 1989、1992 和 1995 年获南京大学理学学士、硕士和博士学位. 现主要研究兴趣为分布计算、通信网及其管理等.

赵生妹　女. 1968 年生于江苏镇江. 于 1989 年和 1992 年分别获南京大学理学学士和中科院上海原子核研究所理学硕士学位. 主要研究方向为多媒体信息系统、软件工程.