一个移动 Agent 算法性能评价的仿真环境*)

李旭晖[1] 何炎祥[1] 曹建农[2]

(武汉大学计算机学院 武汉430072)[1] (香港理工大学电子计算学系 香港)[2]

摘 要 移动 Agent 是一种新的分布计算技术,自90年代被提出以来受到广泛关注。研究人员提出了很多基于移动 Agent 的算法,以解决传统分布式计算问题和移动计算中的问题。然而由于移动 Agent 所具有的反应性、自治性、可移动性等的特性使得对移动 Agent 的算法的性能评价成为一个相当复杂的问题。本文使用直接执行仿真(Direct Execution Simulation)的方法建立了一个移动 Agent 系统的仿真环境 MADESE,通过执行移动 Agent 代码来获取其基本数据,并通过仿真计算出移动 Agent 算法在各种环境中的性能。文章讨论了 MADESE 的事件和同步机制以及其系统环境的设计结构,并介绍了其实现原型。

关键词 移动 Agent,直接执行仿真,移动计算,保守同步协议

A Simulation Environment for Performance Evaluation of Mobile Agents

LI Xu-Hui[1] HE Yan-Xiang[1] CAO Jian-Nong[2]

(Computer School of Wuhan University, Wuhan 430072)[1] (Hong Kong Polytechnic University, Dept. of Computing, Hong Kong)[2]

Abstract Mobile agent has attracted researchers in distributed computing for its features such as reactivity, autonomy and mobility. Many mobile agent-based algorithms are proposed to solve different kinds of distributed problems. However, it is those features that result in the problem to evaluate the performance of these algorithms, because the agent's behavior can be dependent with the environment and can migrate freely. In this paper, we propose a direct execution simulation environment called MADESE which consists of a set of simulators. MADESE can monitor and control the process of the agents, simulate execution environment, and generate the events for performance evaluation. The event and synchronization mechanism of MADESE is described in detail, and a prototype of MADESE is also introduced.

Keywords Mobile agent, Direct execution simulation, Mobile computation, Conservative synchronization protocol

1 引言

移动 Agent 是一种新的网络计算技术,自从上世纪90年代中期被提出以来,已经广泛应用于分布并行计算的各个领域,包括并行处理、信息检索、网络管理以及分布式协调和同步控制。移动 Agent 与传统分布式程序的区别在于,它可以在执行时从一个网络节点动态迁移到另一个网络节点继续执行,并且它还可以在执行期间通过与外界环境的交互获取信息以调整具体的执行情况如动态选择迁移的目标节点等。移动 Agent 在保留传统分布式程序的特性外,还具有移动性、自主性、社会性等特性。这些特性使移动 Agent 在网络带宽受限,传送数据量过大等情况下处理分布计算问题时具有相当优势。因此,近年来研究人员提出了各种基于移动 Agent 的算法,以解决一些传统的分布计算问题和移动计算中的各种问题。

作为移动 Agent 算法的基本优劣评价的标准,对算法的性能评价成为设计移动 Agent 算法的一个重要环节[4]。然而由于移动 Agent 本身所具备的特性,如何准确给出其性能评价成为一个难题。

为解决此问题,本文建立了一个基于直接执行仿真方法[1,6,7]的移动 Agent 仿真环境 MADESE(Mobile Agent Direct Execution Simulation Environment)。MADESE 建立在当前较流行的移动 Agent 平台 IBM Aglets 的基础上,能够在局域网内模拟广域网中的移动 Agent 计算环境。在 MADESE 中,待评价的移动 Agent 算法在环境的监控下直接执行,其执行过程中的具体信息通过仿真环境计算和统计,形成在仿真环境中的性能数据。

本文第2节介绍移动 Agent 性能评价的背景情况和直接执行仿真的相关研究情况;第3节描述 MADESE 环境的结构和设计思路;第4节介绍系统的实现情况和系统的试验情况;最后总结全文。

2 背景和相关工作

一般来说,对分布式程序的性能评价方法主要有三种,即运用数学工具分析建模、实际执行算法获取性能数据和对算法做仿真测试获取性能数据。

运用数学工具对于结构较简单的并行算法能够准确地分析其各种性能指标的复杂度,如时间复杂度、吞吐量等等,并能给出这些指标与各种因素之间的关系。但是建立数学模型和推导性能指标的公式常常是一项相当复杂的工作,随着算法的复杂程度增加,用数学方法分析会变得更加困难且易于出错,而且这种方法对于复杂度相同的不同算法很难给出准

*)项目资助:国家自然科学基金重大研究计划(项目编号:90104005),国家教育部骨干教师基金,香港理工大学研究基金(项目号 A-PD54)。**李旭晖** 博士研究生,主要研究领域:分布并行处理、移动计算、形式语义。**何炎祥** 教授,博士生导师,主要研究领域:分布并行处理、移动计算、软件工程。**曹建农** 副教授,博士,主要研究领域:分布并行处理、移动计算、软件工程。

确的比较结果。对于移动 Agent 而言，其迁移性和自主性使其行为更难预测，因而用数学工具分析对于移动 Agent 的性能评价并不适合。

实际执行的方法一般用于在具有固定的执行环境的情况下测试算法性能。但是移动 Agent 的工作环境通常是广域网，无法限定具体的运行节点和指定工作环境。因此这种方法无法适用于移动 Agent 算法的性能测试。

仿真方法是一种分析分布并行系统的复杂特征的有效手段。通过仿真方法可以将算法在一个可控制的环境中运行并加以分析和测试。仿真方法与实际执行方法之间的区别在于仿真情况下算法的执行和对算法指令的分析处于仿真环境的实时控制中。算法在根据具体环境特性构造的仿真环境下执行可以模拟实际执行时的主要特点，因而获取的性能数据与实际情况非常相似，可以作为对算法评价的可靠依据。仿真方法一般分为事件驱动(event driven)、跟踪驱动(trace driven)和执行驱动(execution driven)三种，其中执行驱动仿真常用于对分布并行系统分析及算法的性能测试。直接执行仿真方法即是典型执行驱动仿真方法。

直接执行仿真方法出现于20世纪90年代初[6]。在这种方法中，仿真器利用已有的系统资源直接执行程序代码，同时模拟那些实际不具备的结构或环境情况，从而模拟出程序在理想环境中执行的情况。早期的直接执行仿真器主要是设计用于模拟高性能计算环境，在这些环境中由于代价的原因无法实际为算法运行提供所有资源，因此采用仿真的方法可以模拟出算法在自由使用资源的情况下的运行状况。比较有代表性的直接执行仿真环境有：

1. LAPSE[7]，用于 Intel Paragon 机器中基于消息传递的并行程序的仿真测试。

2. 并行 Proteus[6]，用于消息传递和共享内存的并行程序的仿真测试。

3. MPI-SIM[5]，用于测试用 MPI 编写的消息传递程序的性能。它可以在运行时获取程序行为信息并仿真实时的运行情况。

传统的直接执行方法运用在分析并行程序性能方面效果显著，MADESE 沿用了其主要思路，并根据移动 Agent 的具体特性做了相应的改进。

3 MADESE 的系统设计

3.1 仿真模型

3.1.1 *概况* 与传统的基于消息传递的分布式程序相比，移动 Agent 具有更多的特性。它可以动态创建、暂停和恢复运行、相互通讯以及在运行时自主迁移。与普通并发程序相似的是，移动 Agent 的程序代码也被与外界交互的操作分割成一系列小的程序代码段，这些通常称为局部代码段(Local Code Block)的代码串行执行并且不与外界交互。因此，移动 Agent 的整个执行过程分为两个部分：执行一个局部代码段；进行一次与外界的交互操作后再执行另一个局部代码段。而仿真工作也自然地分为两个部分：对局部代码段的执行过程的仿真和对交互操作的仿真。

MADESE 的仿真器分为三个部分：宿主系统仿真器、网络仿真器和仿真协调器。宿主系统仿真器模拟移动 Agent 的宿主机器的负荷情况，并计算在模拟的主机环境中执行 Agent 某个代码段所需的时间。网络仿真器负责模拟网络环境，其主要功能是计算出 Agent 进行数据传递和迁移所需的时间。仿真协调器是整个仿真过程的主要部分，它监控各个 Agent 的运行情况，将 Agent 的运行与宿主仿真器和网络仿真器提供的对运行环境的实时仿真信息绑定并负责各 Agent 之间以及 Agent 与仿真环境之间的协调同步。

MADESE 中的仿真过程可以分为三个步骤：

1)代码执行及仿真。各 Agent 在各自的 Agent 系统中执行一个局部代码段后处于阻塞状态，等待仿真器的进一步处理。各 Agent 系统相关的宿主仿真器通过模拟的系统负荷计算出本地运行的 Agent 执行当前代码段所需的虚拟时间。

2)选取交互操作。仿真协调器通过同步协议和其他相关机制，从当前的 Agent 中选取一个子集。

3)被选中的这组 Agent 可以取消阻塞状态并在仿真协调器的控制下转向执行下一个交互操作。交互操作执行完后，重复步骤1，2。

由上可知，仿真系统工作的关键就在于对各 Agent 执行时的交互操作的控制。一般来说，移动 Agent 与外界的交互操作可以分为两种：一种是 Agent 的主体行为，包括 Agent 的通信、暂停、恢复、迁移以及死亡等；另一种则是与环境相关的调用，如调用当前时间、获取当前系统负荷情况等操作。为了精确控制和有效利用这些交互操作，MADESE 采用了事件机制和同步控制机制。

3.1.2 *事件机制* 如果将 Agent 程序中的各局部代码段的执行看作是一组离散状态，则交互操作引发状态变迁从而构成一个离散事件模型。这些由交互操作引发的事件不但是局部代码段的分界点，同时也是 Agent 之间以及 Agent 与仿真环境之间的同步点。此外，通常情况下与这些事件相关的信息是评价 Agent 的性能的主要依据。例如除算法的执行时间外，我们通常需要知道 Agent 发送的消息数和平均消息长度以及发送消息所耗费的时间，Agent 在执行过程中访问的各个网络节点，有时还需要知道一些特定的信息如某段代码的执行时间和某个操作的发生时刻等等。

MADESE 定义了自己的事件模型以支持同步控制和记录运行状况。MADESE 的事件模型由 Agent 行为事件、仿真环境事件两类事件构成。这些事件封装了事件源、发生时间和一些与具体事件类型有关的数据。它们由仿真器和执行环境产生，并且事件的处理在环境的处理下对 Agent 执行过程不构成影响。

Agent 行为事件在 Agent 执行过程中发生。此类事件分为三种：由 Agent 主体行为操作产生的事件、环境相关调用时产生的事件以及用户自定义的事件。如上所述，Agent 的主体行为可以分为六种：创建、通信、迁移、挂起、恢复和死亡。每个 Agent 行为都对应一个或多个 MADESE 事件。如迁移事件中的迁移目标宿主或通信事件中的消息大小等等。MADESE 定义了15个 Agent 主体行为事件类型：

1)两种创建事件，分别在创建开始和结束时产生。

2)四种通信事件，分别在消息发出开始、发出结束、接收开始和接收结束时产生。

3)四种迁移事件，分别在 Agent 的源宿主传送 Agent 代码开始、传送 Agent 代码结束、目标宿主接收 Agent 代码开始、接收代码结束时产生。

4)四种执行事件，分别在 Agent 挂起、挂起恢复、因同步消息阻塞和阻塞恢复时产生。

5)一种死亡事件，在 Agent 被消除时产生。

环境调用事件在 Agent 调用由仿真环境提供的过程时发生。此事件主要用于协调 Agent 与仿真器之间的同步，不包含 Agent 的运行信息。

用户事件在 Agent 执行过程中由程序产生，其内容由程序代码填写，语义由用户指定，而且对事件的分析程序由用户

提供。其作用是使用户可以跟踪特定代码段以获取更多的执行信息。用户事件与传统的验证和评价程序性能时使用的插入测试代码的方法具有相同的功能，但插入测试代码方法的副作用如执行测试代码引起额外的时间开销等不容易消除，而 Agent 通过 MADESE 中的事件机制产生用户事件并监控事件处理的全过程，可将事件处理可能发生的副作用减低至最小。

仿真环境事件包括宿主事件和网络事件两种，分别包括宿主仿真器和网络仿真器的运行信息。这些事件反映了 Agent 执行过程中外界环境的变化情况。仿真环境事件不影响 Agent 的执行，其功能是记录环境信息，使仿真器能通过这些信息能再现 Agent 的运行情况。这是建立可重复的试验环境的基础。

3.1.3 *同步控制机制* 如上所述，移动 Agent 在 MADESE 中执行时存在各 Agent 之间的同步和 Agent 与仿真环境之间的同步，同步点由 Agent 行为事件标志。一般来说，并行仿真器处理并行程序同步问题有两种方法：一种是仿真器在确保不存在其他更早的事件时处理下一事件，此方法称为保守同步协议(Conservative Synchronization Protocol)；另一类则允许不按事件发生时序来处理事件，称为优化协议(Optimistic Protocol)。MADESE 采用基于保守同步协议的方法处理环境中的各种同步问题。

MADESE 中 Agent 与仿真环境之间的同步主要表现在两个方面：一是 Agent 的各行为事件带有时间戳，此时间戳是由宿主仿真器提供的全局虚拟时间；二是 Agent 的环境相关调用需要获取当前虚拟时间的全局环境信息，此时需要与宿主仿真器和网络仿真器同步。

Agent 之间的同步由 Agent 的通信和迁移行为引起，表现在三个方面：

1. 数据发送方发送数据时，因接收方未准备好而引起等待；

2. 数据接收方接收数据时，因暂时无发送方发送消息而引起等待；

3. 数据接收方收到多个数据发送方发送数据时，需要对发送请求做取舍。

前一类同步问题由宿主仿真器和网络仿真器解决。宿主仿真器跟踪本地的每个 Agent，获取当前事件发生时的实际执行时间并计算出虚拟时刻作为时间戳的值。在遇到环境相关调用时，宿主仿真器和网络仿真器模拟出在当前虚拟时刻的环境情况。

后一类同步问题需要各仿真器协调解决。以 Agent 通信为例，数据发送方在遇到发送调用时，根据本地宿主仿真器给出的虚拟时刻加上网络仿真器计算的传送延迟形成发送请求到达接收方的虚拟时刻，发送方的仿真协调器向接收方仿真协调器发出此虚拟时刻，并等待接收方仿真器的应答消息。如果得到允许发送的应答消息则发出消息发送开始事件，并开始传送数据；若收到暂时不能响应发送请求的回答则由发送方发出消息同步阻塞事件，并等待接收方的响应。接收方在准备接收时如果未发现发送请求，则发出消息同步阻塞事件，等待发送方的发送请求；如果接收方在接收时发现有多个发送请求形成的队列，响应请求到达时刻最小的发送请求，此时发送方发出阻塞恢复事件和消息发送开始事件，接收方发出消息接收开始事件，并开始数据传递。数据传递完后，分别由双方的宿主仿真器和网络仿真器计算时间并各自发出消息传递结束事件。Agent 迁移引起的同步处理与之相似。

3.2 系统结构

MADESE 的系统环境由用户界面、仿真控制、仿真系统和执行系统四个模块组成。系统中的主要组成部分如图1所示。

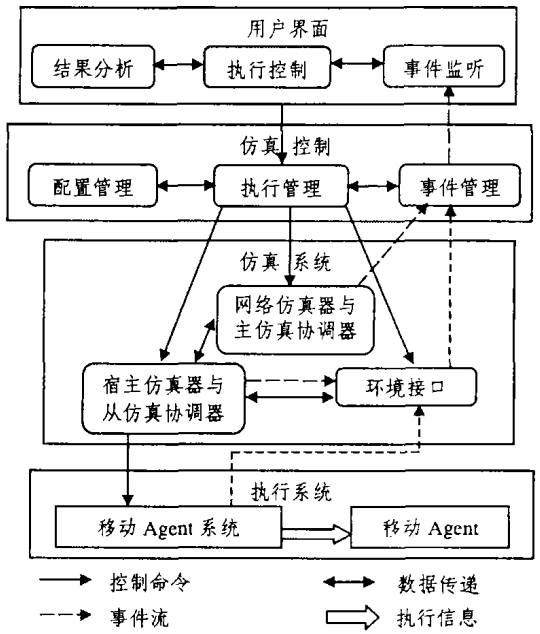

图1 MADESE 的结构

3.2.1 *用户界面与仿真控制* 该模块负责配置仿真环境及任务分配、接收并响应事件以及分析实验结果。

用户通过界面操作配置仿真环境，并将待执行的任务分配到指定节点。配置工作完成后通过执行控制子模块开始进行仿真。在仿真实验中，为了获得可靠的结果需要多次重复实验，而比较不同算法的实验结果只有当算法运行在相同或相似的仿真环境下时才有意义，因此构造可再现的仿真环境是仿真工作的基础。在 MADESE 中系统采用了日志方法来重构执行环境。宿主仿真器和网络仿真器在运行时将环境信息通过环境事件记录，在执行结束后由分析器分析环境事件并形成工作日志或配置文件。用户在仿真实验时，通过配置管理子模块指定相应的日志记录供两个仿真器重构实验环境。

系统执行过程中，各宿主仿真器、网络仿真器、仿真协调器以及移动 Agent 系统均会产生并传送事件。这些事件由事件管理器统一接收，并转发到用户界面中由系统或用户指定的实时事件监听器中，用户通过这些事件监听器可以直接监控 Agent 的运行状况和了解环境变化情况。Agent 算法的性能由结果分析模块完成，MADESE 提供标准的结果分析功能，能通过 Agent 行为事件中包含的数据计算出算法的主要性能指标，如执行时间、迁移情况、消息传递情况等等。同时 MADESE 为用户提供了标准接口，使用户可以通过自己的程序分析事件进行性能评价。

3.2.2 *仿真系统与执行系统* 仿真系统由网络仿真器、各宿主仿真器和各仿真协调器组成。这些仿真器合作完成执行环境的仿真、Agent 执行的同步控制以及执行过程中产生各种事件。仿真系统的运行机制如上节所述。

执行系统负责移动 Agent 的具体执行。移动 Agent 系统在宿主仿真器和管理本 Agent 系统的从仿真协调器的监控下运行指定的移动 Agent。传统的并行程序仿真环境中各程序执行的节点固定，因而需要模拟的运行节点也是固定的。但移动 Agent 执行时的运行环境动态的，Agent 可以在 Internet 中自主决定下一个迁移的目标节点，传统的环境仿真方法不

再适用。MADESE 采用了灵活的执行系统管理机制来处理这个问题。初始情况下系统根据用户指定的配置情况和 Agent 数目建立足够的移动 Agent 系统，在 Agent 执行过程中，网络中的移动 Agent 信息仅由网络仿真器模拟，信息由环境接口提供给移动 Agent。当 Agent 根据需要迁移到新的节点时，由执行控制模块临时创建新的移动 Agent 系统或者将当前未被使用的移动 Agent 系统改名后作为迁移目标节点使用。在此过程中，网络仿真器、主仿真协调器和环境接口合作完成 Agent 系统的名字和实际地址的绑定，并同时更新仿真环境信息。

4 系统实现与测试

基于以上的设计思路，我们实现了一个 MADESE 原型系统 SimulAgent。SimulAgent 系统使用 Java 语言实现，开发工具为 JDK1.3，底层运行的移动 Agent 系统是 IBM Aglets 2.0。根据仿真系统的需要，我们对 Aglets 平台部分做了一定修改，使之能在仿真器的监控下运行 Aglet 程序。系统的主窗口如图2所示。

为测试原型系统仿真环境的功能，我们选取了一些移动 Agent 算法实现作为测试用例进行测试。例如，图3给出了使用移动 Agent 解决负载平衡问题的算法[3]实现在 SimulAgent中运行后通过对事件分析所得到的实验结果。此结果与文[3]中给出的原实验结果相近，说明 SimulAgent 原型构建的仿真环境能有效地测试算法性能。但是，从当前实验结果来看，我们构造的原型系统还有一些不足，主要表现在：

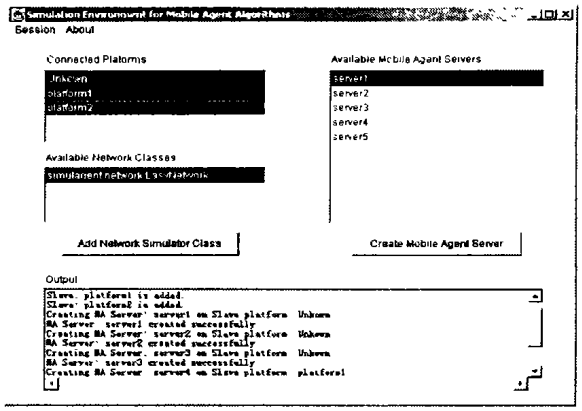

图2 MADESE 界面

1. 宿主仿真器和网络仿真器的仿真能力不够。作为 MADESE 设计的仿真环境模型，SimulAgent 的工作主要集中于实现事件和同步机制，使系统能正确运行移动 Agent 程序并获取数据。宿主仿真器和网络仿真器的实现思路较简单，应用范围较窄，这些问题将会在以后工作中逐步解决。

2. SimulAgent 原型系统在环境相同时用于比较各种算法效果良好，能准确获得算法的具体执行时间以及其他参数。但是对于异构环境下由于运算速度和平台的不同，系统对 Agent 执行时间的计算不够准确，需要收集更多的数据并进一步改进。

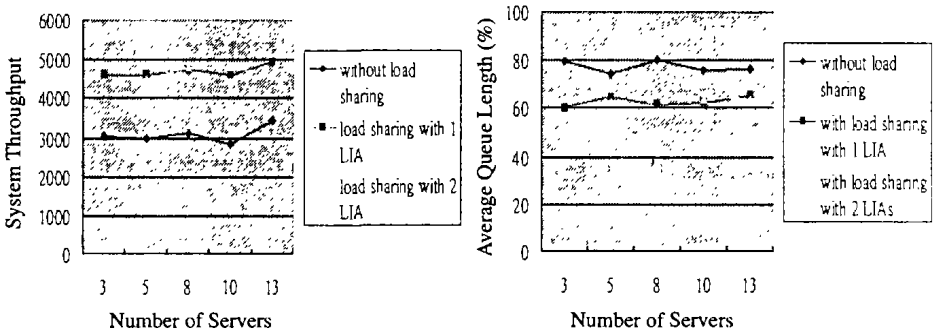

图3 移动 Agent 算法在原型系统中运行的实验结果

结论 本文介绍了一个利用直接执行仿真方法构造测试移动 Agent 算法性能的系统仿真模型 MADESE。MADESE 根据移动 Agent 的特性，建立了一套完整的事件机制和基于保守同步协议的同步机制，使仿真环境能够精确地控制移动 Agent 的运行与同步。在此基础上，文章给出了建立 MADESE 仿真环境的结构，并以此实现了系统原型 SimulAgent。根据原型上的测试结果可知，MADESE 仿真环境在测试移动 Agent 算法性能上是有效的。目前作者正在改善原型系统的仿真性能，使其能更好地模拟大规模网络中移动 Agent 的运行状况。

参 考 文 献

1 Cao J, Bennett G, Zhang K. Direct Execution Simulation of Load Balancing Algorithms with Real Workload Distribution. The Journal of Systems and Software, 2000, 54: 227～237

2 Cao J, Pole M. A Software Environment for Simulating Distributed Task-Scheduling algorithms. Software-Concepts and Tools, 1997, 18: 125～136

3 Cao J, Wang X, Das S K. A Framework of Using Cooperating Mobile Agents to Achieve Load Balancing in Distributed Web Server Groups. In: Proc. 5th Intl. Conf. on Algorithms and Architectures for Parallel Processing (ICA3PP2002) (IEEE Computer Society Press), Oct. 2002

4 Ismail L, Hagimont D. A Performance Evaluation of the Mobile Agent Paradigm. Conference on Object-Oriented, 1999. 306～313

5 Prakash S, Deelman E, Bagrodia R. Asynchronous Parallel Simulation of Parallel Programs. Software Engineering, 2000, 26 (5): 385～400

6 Covington J R, Dwarkadas S, Jump J, Madala S, Sinclair J. The Efficient Simulation of Parallel Computer Systems. International Journal in Computer Simulation, 1991, 1(1): 31～58

7 Dickens P M, Heidelberger P, Nicol D M. Parallelized Direct Execution Simulation of Messasge-Passing Parallel Programs. IEEE Transactions on Parallel and Distributed Systems, 1996, 7 (10): 1090～1105