文章编号:1001-506X(2012)04-0813-07

面向并行 Agent 仿真的合成基准测试模型

余文广,王维平,侯洪涛,李 群
(国防科学技术大学信息系统与管理学院,湖南 长沙 410073)

摘 要:为了评估并行仿真算法的性能,需要建立一个基准测试模型。针对并行 Agent 仿真研究领域中缺乏一种与应用无关的基准测试模型这一问题,在借鉴并行离散事件仿真中经典的合成测试模型 PHOLD 设计思想的基础上,根据基于 Agent 仿真的特点,提出面向并行 Agent 仿真的合成基准测试模型,利用该模型可以方便地合成符合不同应用特点的计算负载,去除与应用相关的因素对性能分析的影响,能够为不同的并行 Agent 仿真研究者提供一个公共的测试基准。最后,采用该模型从实验层次上分析了 Agent 计算粒度、所采用的处理器数目等因素对并行 Agent 仿真加速比的影响。

关键词:并行 Agent 仿真;PHOLD 模型;基准测试模型;性能分析
中图分类号:TP 391.9 **文献标志码**:A **DOI**:10.3969/j.issn.1001-506X.2012.04.31

Synthetic benchmark model for parallel agent-based simulation

YU Wen-guang, WANG Wei-ping, HOU Hong-tao, LI Qun
(*College of Information Systems and Management, National University of Defense Technology, Changsha 410073, China*)

Abstract: In order to evaluate the performance of parallel simulation algorithms, there is a need for a benchmark model. To solve the problem that there is currently lack of such a common benchmark model that is independent of applications in the parallel agent-based simulation (PABS) research community, based on the design principles of parallel HOLD which is a classic synthetic benchmark model for parallel discrete event simulations (PDES), a common benchmark model for PABS is proposed according to the characteristics of agent-based simulations (ABS). This model can easily synthesize various required workloads based on application characteristics and exclude the impact of elements related to specific applications on the performance analysis so as to provide a common benchmark for different PABS researchers. Finally, with this model, the impact of the computation granularity of agents and the number of processors on the speedup is analyzed experimentally.

Keywords: parallel agent-based simulation (PABS); parallel HOLD (PHOLD) model; benchmark model; performance analysis

0 引 言

基于 Agent 的建模仿真(agent-based modeling and simulation, ABMS)方法作为一种自底向上地研究复杂系统的有效途径[1],在社会学、经济学以及作战分析等众多学科领域中得到广泛应用。应用 ABMS 方法研究一些大型复杂系统时,Agent 数量可达 $10^4 \sim 10^6$ 规模[2],这对仿真运行速度提出了巨大的要求,传统的单机串行执行方式很难满足这种需求。并行 Agent 仿真(parallel agent-based simulation, PABS)[3-5]通过将 Agent 仿真模型分配到多个处理单元上同时运行来实现仿真运行加速,是提高基于 Agent 的仿真(agent-based simulation, ABS)运行速度的一个直接而有效的手段。

为了评估时间同步、负载均衡等并行仿真算法性能,需要一个基准测试模型。当前,在并行 Agent 仿真研究领域中,通常是利用一些有代表性的应用作为测试基准,如文献[6]提出采用 Mood Diffusion 模型、Game of Life 模型和 Segregation 模型作为测试基准用例,对 ABS 在图形处理器

收稿日期:2011-07-11;**修回日期**:2012-02-22。
基金项目:国家自然科学基金(60974073,60974074)资助课题
作者简介:余文广(1984-),男,博士研究生,主要研究方向为基于 Agent 的建模仿真、并行仿真。E-mail:wg.yu@126.com

(graphics processing unit, GPU)上的运行性能进行评估;文献[7]提出采用 Boids 模型对其建立的多 Agent 系统仿真中的乐观时间同步协议 DTRD(decision the oretic read delay)的性能进行分析;文献[8]提出采用 Game of Life 模型和 Leadership 模型对 ABS 在多核中央处理器(central processing unit, CPU)、多 GPU 集群上的运行性能进行评估;文献[9]提出采用 Packet-World 模型[10]对其建立的面向多 Agent 系统的时间管理策略进行评估;文献[11]提出采用无人机群体性仿真模型对其建立的减少 Agent 之间通信开销的策略进行实验评估。以上这些模型都是典型的 ABS 应用,但作为测试用例,一方面它们与具体应用相关,应用的特点决定了它们的适用范围,从而可能出现同一个算法采用不同测试用例得出的实验结果大相径庭的情况,降低了研究成果的公信度,如文献[6]采用的测试用例都是逻辑很简单的元胞自动机类型的 ABS 应用,这类应用能够比较有效地映射到 GPU 上运行,测试结果可达近 70 倍的加速比,但如果采用 Agent 逻辑关系比较复杂的 ABS 应用作为测试用例,如文献[11]中采用的无人机协同仿真,未必就会得到如此显著的加速比;另一方面,由于这些测试用例是具体的 ABS 应用,其行为逻辑、计算特点是一定的,对 Agent 计算负载、Agent 交互方式等实验因素不易调节。基于以上所述,对于 PABS 研究领域,我们期望建立一个独立于具体应用的基准测试模型,该模型能涵盖 PABS 中的基本操作,通过调整模型参数可以方便地合成实验所需的计算负载,从而能够去除与具体应用相关的因素对性能分析结果的影响,为不同的研究者提供一个通用的测试基准。

PHOLD 模型[12]是并行离散事件仿真(parallel discrete event simulation, PDES)研究领域中一个经典的合成测试模型,得到广泛应用,但由于 ABS 和离散事件仿真(discrete event simulation, DES)在抽象建模思想、仿真执行策略等方面的不同,PHOLD 模型并不能直接用于评估 PABS 中并行算法的性能。目前,PABS 研究领域中仍缺乏一种独立于具体应用的合成基准测试模型。

本文分析了 PHOLD 模型不适用于 PABS 的原因,在借鉴 PHOLD 模型基本设计思想的基础上,根据 ABS 的特点,建立了面向 PABS 的合成基准测试模型,然后利用该模型,对 Agent 计算粒度、所采用的处理器数目等因素如何影响 PABS 加速比进行了实验分析,展示了如何使用该合成基准模型进行 PABS 的性能分析。

1 PHOLD 模型

PHOLD 模型是经典的 PDES 仿真引擎性能测试基准模型,在 PDES 研究领域中得到广泛地应用。虽然 PHOLD 模型主要针对离散事件仿真范式建立,并不完全适合于 ABS,但其一些如参数设计、负载表示等基本设计思想可以作为借鉴,来建立面向 PABS 的合成基准测试模型。

1.1 PHOLD 模型简介

PHOLD 模型来源于 HOLD 模型,是 HOLD 模型[13]的并行版本(parallel HOLD)。HOLD 模型主要用来考察串行离散事件仿真系统的事件队列操作性能,在该模型中,从事件队列中取出一个事件来执行的唯一结果就是产生一个新的事件并重新插回事件队列(这种操作即称为 HOLD 操作)。PHOLD 模型继承了 HOLD 模型的基本思想,涵盖了 PDES 系统在事件调度、通信、同步等各方面的操作,被广泛用于 PDES 研究领域中各种时间同步、负载平衡等算法的性能测试。

在 PHOLD 模型中,设仿真由 N_{Nodes} 个计算节点组成,每个节点上有 N_{LPs} 个逻辑进程(logic process, LP),每个 LP 的初始事件队列中包含 M 个事件。仿真运行时,每个事件在处理过程中都会调度一个新的事件产生,根据 HOLD 模型的基本思想,整个仿真运行过程中的事件数量保持不变,即 $N_{\text{Nodes}} \times N_{\text{LPs}} \times M$。产生新事件的时戳为 $LVT + Lookahead + \Delta T$,其中,$LVT$ 是该 LP 所在节点的本地仿真时钟;$Lookahead$ 是该 LP 的时间前瞻量;ΔT 为服从某种分布的时间值,通过设定 ΔT 的取值分布可以控制 LP 调度新事件的时间跨度。每个事件的接收者可以是随机的,也可以根据某种规则事先确定。PHOLD 模型中有多个参数可以设置,通过改变参数设置来模拟不同仿真模型的行为。

1.2 PHOLD 用于 PABS 的适用性分析

当前,ABMS 方法更多地是作为一种抽象建模的思想,在仿真执行层并没有像 DES 那样有一个明确、统一的范式,在一些具体应用中,根据建模目的、抽象程度不同,可采用不同仿真执行方式(基于离散事件或基于时间步长)。从 ABMS 基本思想出发,在一般意义上探讨 PHOLD 模型用于 PABS 性能测试的适用性。从以下三个方面进行分析:

(1) 仿真调度

PHOLD 模型是面向 PDES 提出的,采用的是基于离散事件的仿真调度方式。在这种方式下,处理当前发生的事件时,需要安排未来时刻发生的事件,然后根据未来事件队列中时戳最小的事件来推进仿真时钟。而在 ABMS 中,Agent 是自治实体,即使没有外界输入事件,Agent 也能主动地产生改变自身和环境状态的输出事件——Agent 行为事件的不可预测性,对于 ABS 而言,更多是采用基于时间步长的调度方式,以 Agent 为基本的调度单元,由 Agent 的主动行为推进仿真时钟。

(2) HOLD 操作

PHOLD 模型的一个基本思想就是 HOLD 操作,仿真

运行过程中系统事件队列的规模保持不变。在基于 Agent 的仿真中,并不是通过调度事件来推进仿真时钟,而且 Agent 能主动产生行为事件,事件的触发也不一定是一对一的关系。因此,将 HOLD 操作作为 ABS 中的一个基本操作并不合适。

(3) 模型表达能力

在 DES 中,模型的行为可以认为是调度事件并对事件进行处理,因此,在 PHOLD 模型中,通过调整事件触发和生成规则以及事件处理负载可以模拟不同行为特点的 DES 模型。在 ABS 中,Agent 模型能主动地从环境中感知信息,并根据这些信息以及自身的行为规则进行思考决策,然后付诸行动,PHOLD 模型无法描述 Agent 的这些行为特点,利用 PHOLD 模型对基于这些特点而提出的 PABS 并行策略进行测试缺乏有效性和说服力。

从以上分析可以看出,PHOLD 模型并不适合作为 PABS 的基准测试模型,需要根据 ABS 的特点,建立适合于 PABS 的合成基准测试模型。

2　PABS 性能测试基准模型

2.1　Agent 概念模型

Agent 概念在人工智能、软件工程等不同学科领域内有不同含义,从建模仿真实践的角度出发,一个典型的 Agent 应该具备自治性、完备性和社会性[1],其行为过程可以由 sense-think-act 行为范式来表示[14],即它能够主动地从其生存的环境中获取感知信息,并根据这些信息以及自身的行为规则、知识体系进行判断、决策,然后产生改变自身、环境甚至其它实体状态的行动。一般性的 Agent 概念模型如图 1 所示。

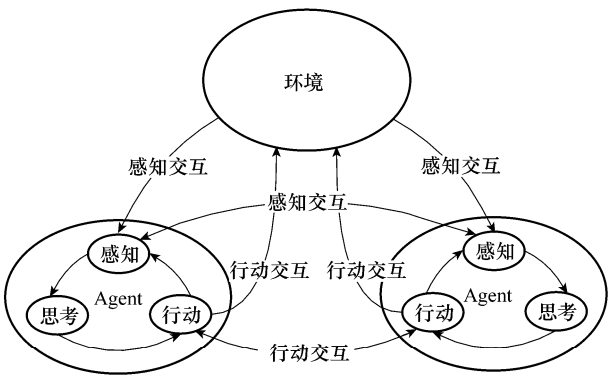

图 1　Agent 概念模型

根据图 1,可以将 Agent 仿真模型抽象为一个二元组

$$M=\langle A,\ E\rangle$$

式中,A 表示 Agent 集合,E 表示环境。

每个 Agent 可用一个四元组来定义

$$a=\langle Info,\ S_a,\ Act,\ \delta_a\rangle$$

式中,$Info$ 代表 Agent 感知到的信息集合,可表示为 $Info=Info_r\cup Info_d$,其中,$Info_r$ 表示从其它 Agent 接收到的信息,$Info_d$ 表示 Agent 通过自身探测而得到的信息;S_a 代表 Agent 的状态集合;Act 代表 Agent 所产生的行动集合,可表示为 $Act=Act_s\cup Act_I$,其中,Act_s 表示作用于 Agent 自身的行动(如移动),Act_I 表示 Agent 所产生的交互行动(如发送交互信息);$\delta_a:Info\times S_a\rightarrow Act\times S_a$,表示 Agent 通过 think 的行为功能映射。

环境 E 可以用一个三元组表示

$$E=\langle S_E,\ X,\ \delta_E\rangle$$

式中,S_E 表示环境的状态集合;X 表示外部更新事件集合;$\delta_E:S_E\times X\rightarrow S_E$,表示环境状态更新映射。

环境是一个 n 维有界或无界的度量空间,为 Agent 提供生存空间。环境有多种不同的类型,参考 RepastHPC[15] 中的定义,可以是网络、格网或欧几里得连续空间。不同类型环境会影响 Agent 的行为方式。

2.2　面向 PABS 的合成基准测试模型

根据 2.1 节描述的 Agent 概念模型,参考 PHOLD 模型,建立面向 PABS 的合成基准测试模型。该模型可设置的参数如表 1 所示,其中,"$Simulation::$"、"$Environment::$"和"$Agent::$"为域名标志符,分别表示其后的参数属于仿真设置、环境模型和 Agent 模型的参数。

表 1　PABS 基准测试模型参数

参数	说明
$Simulation::N_{\mathrm{LPs}}$	仿真采用的逻辑进程数目
$Simulation::N$	Agent 的数目,仿真运行过程中不变
$Simulation::\Delta T_{\mathrm{step}}$	仿真步长
$Simulation::T_{\mathrm{End}}$	仿真结束时间
$Environment::E_{\mathrm{ty}}$	环境的类型,有 Network、Grid 和 Space 三种类型
$Environment::E_{\mathrm{sc}}$	描述环境范围的结构体,若结构体为空,则表示无界
$Agent::R_S$	Agent 的感知距离
$Agent::\rho_s$	Agent 的感知概率
$Agent::L_T$	Agent 对感知事件处理的计算负载
$Agent::\lambda_L$	Agent 计算负载所服从的分布,L_T 和 L_{AS} 均服从该分布
$Agent::R_I$	Agent 的交互距离
$Agent::\rho_I$	Agent 的交互概率
$Agent::L_{\mathrm{AS}}$	Agent 处理作用于自身行动的计算负载
$Agent::\lambda_T$	Agent 产生交互事件的时戳所服从的分布

该基准测试模型中包含 N 个 Agent 和 1 个环境模型,采用基于时间步长的仿真调度策略来执行该测试模型。在每个仿真步长,调度所有的 Agent 执行其当前时刻的行为,然后根据各 Agent 计算的结果更新环境,并推进到下一个仿真时刻。Agent 的行为过程和环境更新如下所述:

(1) Agent 行为过程

Agent 的行为采用 sense-think-act 范式来描述,即从环境中感知信息,然后根据这些信息和自身的行为规则进行

思考决策,最后付诸行动。每个 Agent 都有一个感知列表和交互列表,分别用于存放感知事件和交互事件。在Agent 感知环境时,获得一则关于环境的消息就会产生一个相应的感知事件;交互事件产生于两个 Agent 发生交互时,包含发起交互的 Agent 想要通知目标 Agent 的信息。交互事件由发起交互的 Agent 产生,最终存储在目标 Agent 的交互列表中。感知事件和交互事件都会在"think"阶段被处理。sense-think-act 行为过程具体为:

① sense 阶段主要产生感知事件。感知事件的产生有两个来源:一部分从其它 Agent 处获得,即在该 Agent 交互列表中存放的事件;另一部分则在其感知距离内,根据感知概率由其自身探测获得。这两类事件都存放在感知列表中。

② think 阶段主要对每个感知事件进行处理。对每个感知事件进行处理的计算负载可以由一些基本的算术或逻辑运算来合成,如对每个事件的处理用执行 m 遍矩阵相乘来表示。

③ act 阶段主要产生并执行行动事件。一部分行动作用于 Agent 自身,为了方便表示,将这部分行动事件归结为一个事件,其处理负载为 L_{AS} ;另一部分行动用于交互。Agent 在其交互范围内,根据交互概率确定与哪些 Agent 发生交互,然后向它们发送交互事件。每个交互事件的时戳为 $T_{\text{current}}+\Delta T$,其中, T_{current} 为当前仿真时刻; ΔT 为服从 λ_T 分布的一个随机数,为仿真步长的整数倍。执行完上述行动事件后,Agent 移动并更新自身的位置等状态信息。

(2) 环境更新

在该基准测试模型中,环境模型拥有一个存放各 Agent 信息的数据结构,每个 Agent 通过访问该数据结构来获取环境信息。为了确保在同一仿真时刻不同 Agent 访问的是同一数据结构,在所有 Agent 当前仿真时刻行为执行完毕之后,再根据 Agent 计算结果统一更新该数据结构。

基于以上所述,PABS 基准测试模型仿真执行过程如图 2 所示。

```
1 //创建 N 个 Agent,设置其初始状态
2 InitAgents();
3 //根据各 Agent 初始状态,初始化环境模型
4 InitEnvironment();
5 //初始化仿真时钟
6 double T = 0.0;
7 while(T < T_End ){
8     for each Agent In Agents do {
9       从交互队列取出当前要处理的事件,放入集合 SenseEvents;
10      将该 Agent 感知距离 R_S 内的 Agent 放入集合 Agents_D ;
11      for each Agent In Agents_D do {
12          产生(0,1)之间均匀分布随机数 rand;
13          if(rand < ρ_S ) then {
14              产生一个关于该 Agent 的感知信息,加入集合 SenseEvents;
15          }endif
16      }endfor
17      for each SenseEvent In SenseEvents do {
18          执行处理该感知事件的计算负载 L_T ;
19      }endfor
20      执行作用于自身的行动事件的计算负载 L_AS ;
21      将该 Agent 交互距离 R_I 内的 Agent 放入集合 Agents_I ;
22      for each Agent In Agents_I do {
23          产生(0,1)之间均匀分布随机数 rand;
24          if( rand < ρ_I ) then {
25              生成交互事件,事件时戳为 T+ΔT ;
26              将交互事件写入目标 Agent 的交互队列中;
27          }endif
28      }endfor
29      Agent 移动并更新自身的位置等状态信息;
30  }endfor
31  //根据各 Agent 计算结果,更新环境状态
32  Env.update();
33  T = T + ΔT_step ;
34 endwhile
```

图 2 PABS 基准测试模型仿真执行算法

3 仿真实验分析

3.1 测试模型描述

本节实验所采用的基准测试模型的配置如下:

(1) 环境模型是一个 1 000×1 000 的二维连续空间。

(2) 包含 N 个 Agent,每个 Agent 的感知距离为 10,感知概率为 1,交互距离为 20,交互概率为 1。

(3) N 个 Agent 平均分配到 n 个处理器上并行执行。

(4) 对每个感知事件执行 $L_T\times G$ 遍整数乘法,其中, L_T 服从[1,100]之间的均匀分布;G 用于控制事件处理的粒度。

(5) 仿真初始时刻为 0,仿真步长为 1,仿真结束时间为 100。

(6) Agent 产生交互事件的时戳为当前仿真时刻加上仿真步长。

实验过程中,通过调节 G、N、n 来验证加速比随 Agent 计算粒度、Agent 数目以及所采用的处理器数目的变化关系。

本实验采用的硬件平台是配备了 Intel i7-860 多核处理器和 4G 内存的桌面计算机。该 CPU 采用了超线程技术,具备 4 核 8 线程,可看作是有 8 个同时可用的执行核,每个执行核的时钟频率为 2.8GHz,执行核之间通过共享内存通信。软件方面,选用 OpenMP 作为并行编程,在集成了 Intel Parallel Studio 的 Visual Studio 2005 开发环境中进行仿真应用开发。

3.2　实验分析

本实验分为 4 组,分别用来测试图 2 所示算法的时间复杂度、多核计算平台上线程数目与所利用的处理器核数目的关系、所采用的处理器核心数目对加速比的影响和 Agent 计算粒度对加速比的影响。

(1) PABS 基准测试模型仿真执行算法时间复杂度实验分析

该组实验主要验证图 2 给出的仿真执行算法的时间复杂度,采用串行执行的方式。设定 $G=50$,改变 Agent 的数目 N,执行 100 个仿真步长,Agent 的 sense、think、act 执行时间以及总的仿真执行时间如图 3 所示(以下各图中的数据是利用 Intel Parallel Studio 的 Amplifier 工具获得)。

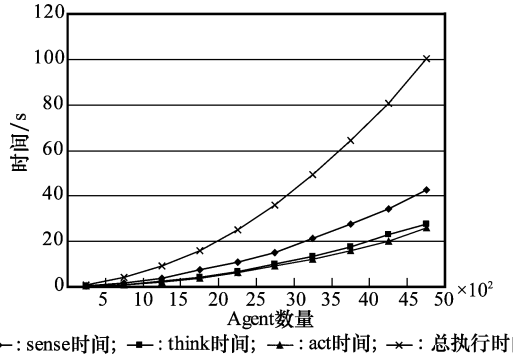

图 3　仿真执行时间随 Agent 数量的变化情况

从图 3 中可以看出,Agent 的 sense、think、act 执行时间占据了绝大部分的仿真执行时间,它们都大致随 N 的增大而成平方关系增长,即时间复杂度为 $O(N^2)$。当环境规模一定时,增加Agent数量后,每个Agent的感知事件和接收到的交互事件数量都会增加,即每个 Agent 在一个仿真步长内的计算量随 N 的增加而增加,执行 N 个 Agent 的时间复杂度大致为 $O(N^2)$,与文献[16]给出的结论一致。由此可知,当仿真应用中 Agent 数量非常多时,仿真时间将会是巨大的,仿真运行速度将是制约大规模 ABMS 应用的重要因素。

(2) 线程数目、计算粒度与处理核数目之间关系的实验分析

多核计算平台通常通过多线程机制将多个处理核利用起来。确定线程被调度到哪个处理核上执行,通常有两种途径:一种是利用某些多线程应用编程接口(application programming interfact, API)(如 win32,OpenMP 不提供此类 API)将线程绑定到某个核上运行;一种是由操作系统全权负责线程调度[17]。出于保证程序在不同硬件平台的可移植性以及不扰乱操作系统本身的优化调度算法考虑,本文实验采用后一种方式。

由于不指定线程在哪个处理核上执行,在有 8 个执行核的硬件平台上,创建 8 个线程并不一定能充分利用 8 个执行核,如图 4(a)所示,在每个线程计算粒度比较小时,CPU 使用率(并行运行的 CPU 核的平均数目)仅为 5.53,而当计算粒度比较大时,如图 4(c)所示,CPU 使用率可达到 7.59,接近理想值 8。由于线程是软件层次上并行执行的基本单元,创建的线程数目也决定了 CPU 使用率的最大值,如图 4(b)所示,在整体计算规模和图 4(c)中一样的情况下,创建 4 个线程,CPU 使用率是 3.92,不超过 4。

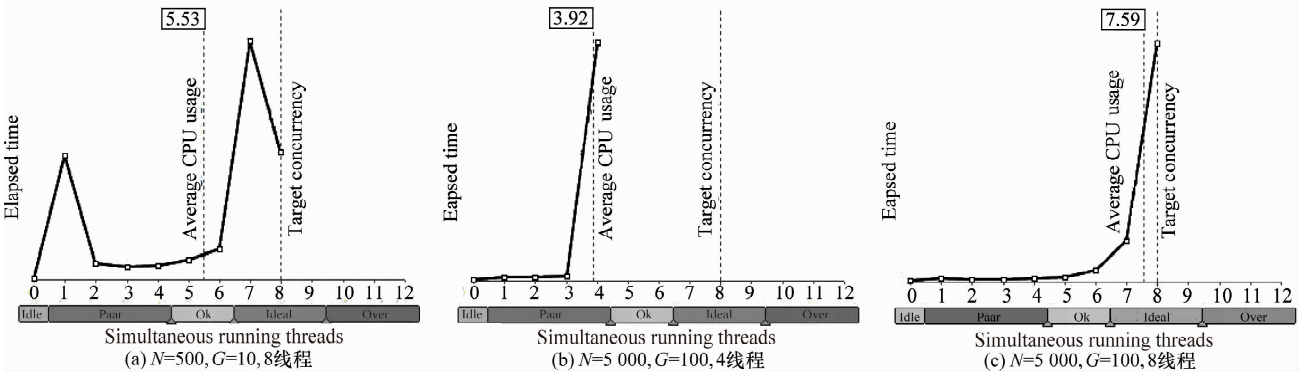

图 4　线程数目、计算粒度与 CPU 平均使用率间的关系

(3) 加速比与所采用的处理核数目之间关系的实验分析

本组实验主要考察加速比随采用的处理核数目增加的变化情况,其中,设定 $N=5\ 000$、$G=100$。从上一组实验分析中可知,由于不为每个线程指定处理核,并不能直接指定并行仿真所用的处理核数目,但只要每个线程的计算粒度足够大,所采用的线程数目和 CPU 核心数目(用 CPU 使用率衡量)还是很接近的。图 5(a)显示了采用不同线程数目对应的 CPU 使用率。由于总的计算量不变,随线程数目的增加,每个线程分得的计算量有所下降,CPU 使用率与线程数目的差距有所扩大,但整体上还是可以认为采用的处理核数目约等于线程数目,图 5(b)中横轴"处理核数目"实际上就是采用的线程数目。在本实验中,并行开销主要是同步开销和跨线程通信开销。具体来说,一是在每个仿真

步长内，必须确保每个 Agent 都执行完毕后才能统一更新环境信息，即每个线程执行完所负责的 Agent 模型实例后需进入栅障同步；二是属于不同线程之间的 Agent 发生交互时，需要对接收消息的 Agent 交互队列进行加锁，以避免发生读写逻辑错误。当线程数目增加时，跨线程通信的消息数量会随之增加，栅障开销也会增加。从图 5(b)中可以看出，随着线程数目的增加，虽然加速比在提升，但其增长率在明显变缓，这表明增加执行核数目所带来的并行开销越来越多地抵消其带来的计算加速。由于本实验采用的平台只有 8 个执行核，根据图 5(b)可以推测，当采用的执行核数目足够大时，再增加执行核数目，很可能会带来加速比的下降。

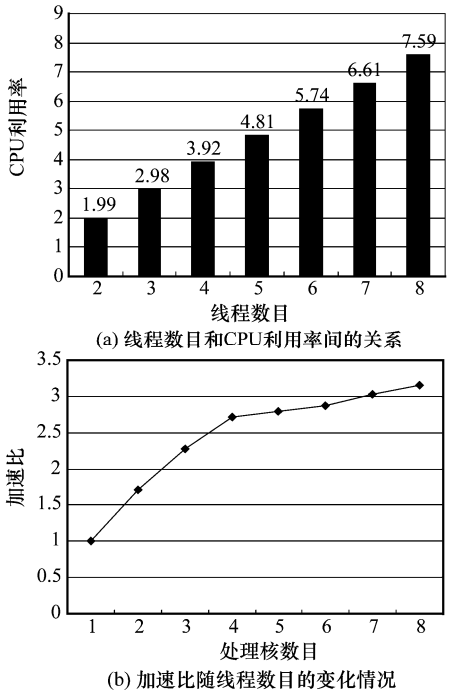

图 5 加速比随采用的处理核数目的变化情况

(4) 加速比与 Agent 计算粒度之间关系的实验分析

本组实验主要验证加速比与 Agent 计算粒度间的关系。设定 $N=5\ 000$、$n=8$。在这里，计算粒度是指计算所花费的时间。由于 Agent 数量和采用的线程数目不变，发生跨线程通信的事件数量以及栅障同步开销也就基本保持不变，则并行开销整体基本保持不变，图 6(a)可以清楚地说明这一点。在并行开销一定的情况下，Agent 计算粒度越大，通过多核并行带来的计算性能提升就越大，加速比也就随之提升。图 6(b)中加速比随 Agent 计算粒度的增加而增加，而且增长率没有明显放缓的趋势，这表明对于计算粒度大的 Agent 仿真应用，通过并行化方式可获得比较好的运行加速。

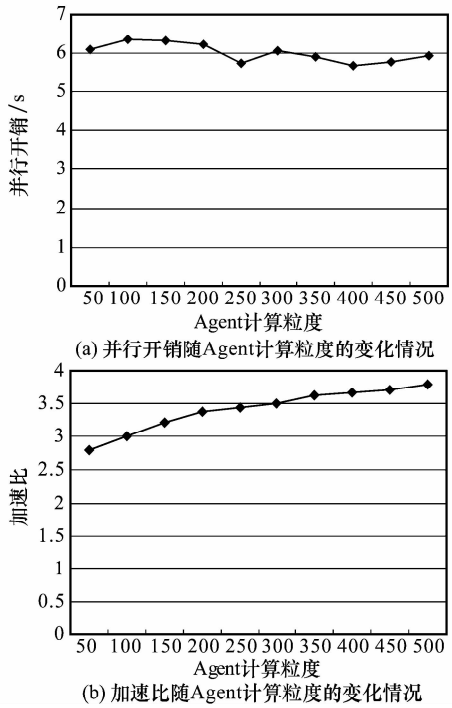

图 6 加速比与 Agent 计算粒度的关系

通过以上 4 组实验，我们可以得到以下结论：(1) PABS 基准测试模型仿真执行时间复杂度为 $O(N^2)$；(2) 在多核 CPU 并行平台上，每个线程计算粒度越大，CPU 使用率越接近所创建的线程数目；(3) 并不是只要增加并行执行所用的处理器或处理核数目就能带来加速比的提升，加速比的增长率有随处理器或处理核数目的增加而减小的趋势；(4) 加速比会随 Agent 计算粒度的增大而增加。此外，从实验中也可以看出，本文提出的合成基准测试模型在探讨并行 Agent 仿真性能中，具有很强的灵活性和通用性，能够去除将具体应用作为测试实例带来的局限性。

4 结 论

为了评估并行 Agent 仿真的性能，本文在借鉴 PHOLD 模型参数设计、负载设计等方面的基本思想基础上，根据 ABS 的特点，提出一个面向 PABS 的合成基准测试模型。该模型独立于具体应用，可以通过调整模型参数，方便地合成实验所需的计算负载，能够去除与具体应用相关的因素对性能分析的影响，在 PABS 领域中建立起类似于 PDES 领域中的 PHOLD 的通用基准测试模型。利用该模型，对 Agent 计算粒度、所采用的处理器数目等因素如何影响加速比进行实验分析，得出一些定性结论。

本文在进行实验分析时，只是粗略地将所有的 Agent 平均分配到各个处理核心上，由于测试用例中假设所有的 Agent 随机平均分布在环境中，而且所有的Agent都是同

一种类型,这样的任务划分在运行过程中使得各个处理核心上的计算负载基本保持平衡。在下一步的工作中,将对 PABS 中的任务划分、负载平衡等策略进行更深入地研究,并利用本文提出的合成基准测试模型对其进行性能评估。

参考文献:

[1] Macal C M, North M J. Tutorial on agent-based modeling and simulation[J]. *Journal of Simulation*, 2010, 4(4): 151 - 162.

[2] Macal C M, North M J, Pieper G, et al. Agent-based modeling and simulation for exascale computing[J]. *SciDAC Review*, 2008(8): 34 - 41.

[3] Popov K, Vlassov V, Rafea M, et al. Parallel agent-based simulation on a cluster of workstation[C]// *Proc. of the 9th International European Conference on Parallel and Distributed Computing*, 2003: 470 - 480.

[4] Gebre M R. MUSE: a parallel agent-based simulation environment[D]. Miami: Miami University, 2009.

[5] Cosenza B, Cordasco G, Chiara R D, et al. Distributed load balancing for parallel agent-based simulations[C]// *Proc. of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing*, 2011: 62 - 69.

[6] Perumalla K S, Aaby B G. Data parallel execution challenges and runtime performance of agent simulations on GPUs[C]// *Proc. of the Spring Simulation Multi-Conference*, 2008: 116 - 123.

[7] Lees M, Logan B, Theodoropoulos G. Using access patterns to analyze the performance of optimistic synchronization algorithms in simulations of MAS[J]. *Simulation*, 2008, 84(10 - 11): 481 - 492.

[8] Aaby B G, Perumalla K S, Seal S K. Efficient simulation of agent-based models on multi-GPU and multi-core clusters[C]// *Proc. of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010.

[9] Helleboogh A, Holvoet T, Weyns D, et al. Extending time management support for multi-agent systems[C]// *Proc. of the Joint Workshop on Multi-agent and Multi-agent-based Simulation*, 2005: 37 - 48.

[10] Weyns D, Helleboogh A, Holvoet T. The packet-world: a test bed for investigating situated multi-agent systems. Unland R, Klusch M, Calisti M. *Software agent-based applications, platforms, and development kits*[M]. Basel: Birkhauser Verlag, 2005.

[11] Jang M W, Agha G. Agent framework services to reduce agent communication overhead in large-scale agent-based simulations [J]. *Simulation Modelling Practice and Theory*, 2006, 14 (6): 679 - 694.

[12] Fujimoto R M. Performance of time warp under synthetic workloads[C]// *Proc. of the SCS Multiconference on Distributed Simulation*, 1990: 23 - 28.

[13] Vaucher J G, Duval P. A comparison of simulation event list algorithms[J]. *Communications of the ACM*, 1975, 18(4): 223 - 230.

[14] Hybinette M, Kraemer E, Xiong Y, et al. SASSY: a design for a scalable agent-based simulation system using a distributed discrete event infrastructure[C]// *Proc. of the Winter Simulation Conference*, 2006: 926 - 933.

[15] Collier N T, North M J. Repast SC++: a platform for large-scale agent-based modeling. Dubitzky W, Kurowski K, Schott B. *Large-scale computing techniques for complex system simulations*[M]. New York: Wiley Press, 2011.

[16] Sanchez S M, Lucas T W. Exploring the world of agent-based simulations: simple models, complex analyses[C]// *Proc. of the Winter Simulation Conference*, 2002: 116 - 126.

[17] 苏年乐. 仿真模型可移植性规范的多核并行化研究[D]. 长沙: 国防科学技术大学,2010. (Su N L. Parallelization of simulation model portability specification on multi-core computer[D]. Changsha: National University of Defense Technology, 2010.)