乐观策略下并行离散事件仿真动态负载划分优化算法

张颖星 姚益平
(国防科学技术大学计算机学院 长沙 410073)

摘 要 动态负载划分是提高并行离散事件仿真运行性能的有效途径之一.现有研究往往孤立地考虑计算负载平衡和通信负载优化,使得复杂应用背景下整体性能低下.论文综合考虑仿真模型计算负载和交互模式,提出了一个基于带权重无向图有限容量 k 划分问题的并行离散事件仿真负载划分模型,并配合一套通用的仿真运行性能度量方法,提出了一个基于顶点交换的启发式局部搜索近似划分算法,实现了在计算负载平衡的前提下系统通信负载最优化,其近似解与全局最优解比值不小于$(1-1/|N|)(1-\varepsilon)$.实验证明了该动态负载划分算法的有效性和实用性.

关键词 负载划分;局部搜索算法;并行离散事件仿真;乐观同步策略
中图法分类号 TP391 **DOI 号**: 10.3724/SP.J.1016.2010.00813

A Dynamic Partitioning Algorithm Based on Approximate Local Search for Optimistic Parallel Discrete Event Simulation

ZHANG Ying-Xing YAO Yi-Ping
(*School of Computer, National University of Defense Technology, Changsha* 410073)

Abstract With the rise of simulation platforms which support efficiently migration, the dynamic partitioning mechanism can significantly improve performance of the PDES. How to estimate and optimize the communication structure becomes an essential research topic for dynamic partitioning. Currently, there are several dynamic partitioning algorithms. Since they consider computation and communication load balancing isolated, these algorithms may suffer under complex application background. This paper formalizes the dynamic partitioning problem statement which combine both computation and communication load balancing, and proposes an algorithm based on approximate local search, which obtains a solution with value no smaller than $(1-1/|N|)(1-\varepsilon)$of the optimal solution value in polynomial time complexity. The experiments show that the algorithm has high performance and low overhead.

Keywords partitioning; local search algorithm; parallel discrete event simulation; optimistic synchronization

1 引 言

负载划分(partitioning)和时间同步(synchronization)是决定并行离散事件仿真(PDES)性能的最重要的两个因素[1].由于早期仿真应用规模有限,运行状态相对稳定,动态划分对性能的提升并不明显,此外,由于很多仿真平台对迁移机制支持不足,动态划分的额外开销较大,使得对并行离散事件仿真负载划分的研究长期局限于基于模型行为预测的

收稿日期:2009-06-17;最终修改稿收到日期:2010-04-13.本课题得到国家自然科学基金(60773019)、国家教育部博士点基金(200899980004)资助.**张颖星**,男,1981 年生,博士,主要研究方向为并行离散事件仿真、基于 Agent 仿真等. E-mail: stars.zyx@gmail.com. **姚益平**,男,1963 年生,教授,博士生导师,主要研究领域为并行与分布仿真、虚拟现实等.

静态负载划分[2-4].然而,随着仿真模型越来越复杂,特别是一些研究者提出将并行离散事件仿真支撑技术与 Agent 建模技术相结合以支持复杂系统的研究[5-6],系统负载往往具有显著的动态性,传统的静态划分技术已难以保证仿真的整体运行效率;同时大量支持高效迁移机制的并行离散事件仿真平台的出现也为动态负载划分提供了底层支持[7-9],因此,动态负载划分近年来越来越受到并行离散事件仿真研究者的关注.

动态负载划分的关键问题是仿真运行性能度量和负载迁移策略.早期的动态负载划分算法主要考虑计算负载平衡因素以减少各计算节点间本地仿真时间推进的差异,但随着高性价比的集群系统已逐渐成为大规模并行离散事件仿真的主流硬件平台,通信负载对性能的影响越来越显著,如何在计算负载平衡的前提下根据模型交互模式优化通信负载已成为目前动态负载划分研究的热点.本文综合考虑仿真模型计算和通信负载,将并行离散事件仿真的动态负载划分归结为一个带权重无向图的有限容量 k 划分问题,首先通过模型计算负载和节点计算能力确定各个划分集合的容量,然后采用基于顶点交换启发式的局部搜索近似划分算法,根据模型交互模式对负载划分进行优化,使得在计算负载平衡的前提下通信负载最优化.

本文第 2 节介绍并行离散事件仿真领域近期动态负载划分的相关研究;第 3 节给出动态负载划分的形式化描述;第 4 节详述本文的负载划分优化算法;第 5 节给出测试方法和实验结果;最后总结全文.

2 相关研究

早期的并行离散事件动态划分算法往往只考虑计算负载平衡.Reiher 和 Jefferson[10] 提出的方法是以处理器利用率来评估仿真运行状态,周期性查找过载和欠载的计算节点,若两者间计算负载差距大于规定的阈值,则通过迁移进行负载平衡;Glazer 和 Tropper[11] 提出的算法思想与前者类似,但进一步提出 simuation advance rate 的概念来评估运行状态,同时采用 heuristic bin packing algorithm 进行负载划分,获得了更好的效果;Carothers 和 Fujimoto[12] 也提出一种基于平台信息的动态划分算法,获得了不错的性能提升.这些算法的主要问题在于没有考虑仿真模型通信负载,在仿真模型计算负载较为稳定而交互关系动态性较强的仿真应用中加速效果较差;同时,部分算法对仿真运行性能评估需要底层操作系统或特定仿真平台的支持,限制了其应用范围.

Wilson 和 Shen 在文献[13]中提出一种分段动态划分算法,首先由一个中央节点收集系统运行信息并确定各计算节点的迁移量,然后各节点根据不同的策略来决定具体迁移哪个逻辑进程,而通信负载优化策略是其中的一个选项.该算法引入了通信负载优化的思想,但由于在不同策略中分别孤立地考虑计算负载平衡或通信负载优化,使得在复杂应用背景下两者相互影响,产生大量额外开销,整体性能下降.Slavik 等人[14] 提出的动态划分算法也存在类似的问题,该算法提出了通过带权重的无向图对通信负载进行优化的思想,但没有考虑计算负载平衡的影响,同时该算法提出的仿真运行性能度量方法也难以应用于实际系统.Low 在文献[15]中提出了一个综合考虑计算负载平衡和交互模式优化的模型,获得了不错的效果,但该算法主要针对 BSP-TW 乐观同步协议,限制了其应用范围.Peschlow 等人在文献[16]中提出一种柔性的动态负载划分算法,该算法综合考虑了计算负载平衡和通信负载优化问题,设计了一套通用的并行离散事件仿真运行性能度量方法,但由于采用的迁移算法无法保证收敛,在部分情况下会产生迁移振荡.综上所述,尽管近年来对基于模型间交互模式的动态负载划分技术研究发展较快,但目前还没有一个完善的适用于多种乐观同步协议且综合考虑负载平衡和通信模式的并行离散事件仿真动态负载划分算法.

3 问题模型

我们的问题模型描述是基于通用的并行离散事件仿真“消息-事件”范型:系统由一组逻辑进程 Logic Process(LP)组成,LP 之间通过相互调度事件进行交互,每个 LP 通过处理一系列事件来推进仿真时间,在仿真过程中事件的处理会改变 LP 的内部状态,同时输出新的事件[1].LP 是负载划分的基本单位,可以自由地分配到各个计算节点上运行.每个计算节点通过一个节点控制器(host controller)来管理在该节点上运行的 LP,共用一个事件调度队列,按事件时戳的非递减顺序执行,以保证全局因果序约束(global causality constraint).

负载划分的基本任务即将 LP 合理地分配到各个计算节点上以获得最高的系统运行效率,为达到该目的需要综合考虑系统计算负载平衡和通信负载

优化两方面的因素:计算负载平衡的主要任务是保证 LP 划分集的事件处理计算负载与对应计算节点计算容量相匹配,以减少由各计算节点局部仿真时间(LVT)推进的差异而引起的同步等待和回滚开销[1];通信负载优化的主要任务是根据 LP 事件调度依赖关系将交互频繁的 LP 分配到同一计算节点上,以减少由远程事件调度而引起的回滚开销[14,16].

基于上述"消息-事件"范型,并行离散时间仿真负载划分模型形式化描述如下:设 L 表示逻辑进程集合,E 表示逻辑进程间事件调度关系,首先以仿真系统中所有 LP 为顶点,构造一个全连通的无向图 $G(L,E)$. 定义边权重 $w:L\times L\rightarrow R$ 表示单位仿真时间内任意两个 LP 间相互调度事件数;边权重 $w(l_1,l_2)=0$ 表示 l_1 和 l_2 之间没有交互. 定义点权重 $c:L\rightarrow R$ 表示一个逻辑进程的计算负载. 设 N 表示硬件平台计算节点的集合,$U:N\rightarrow R$ 表示计算节点的计算容量,则负载划分问题可以表示为:从无向图中选出一个边集合 $S\subset E$,使得 $G(L,E-S)$ 分成 $|N|$ 个互不联通的子图(记为 $\{G_1,G_2,\cdots,G_{|N|}\}$),且存在一个映射 $\delta:G_n\rightarrow N$ 满足 $\forall\delta(G_n)\in N:\sum_{l\in G_n}c(l)\leq U(\delta(G_n))$;负载划分优化问题即找到满足 $Min\sum_{e\in S}c(e)$ 的边集合 S,对应映射 $\delta:G_n\rightarrow N$ 即负载划分结果. 该问题是一个 NP 问题[17],本文假设系统中逻辑进程计算负载符合泊松分布,则我们首先根据各个计算节点的容量和各个 LP 的计算负载,在满足系统计算负载均衡的条件下确定划分集合容量,然后给出一个基于顶点交换的启发式局部搜索近似划分算法,其近似解与全局最优解的比值不超过 $(1-1/|N|)(1-\varepsilon)$.

4　动态划分算法

根据上述基本问题模型描述,我们的算法主要包含以下几个部分:(1)初始化仿真系统,根据系统基本信息将 LP 划分到各个计算节点上;(2)通过各计算节点上的仿真控制器对运行信息进行采样;(3)设置一个独立的控制节点周期性收集系统的运行信息,从计算负载和通信负载两方面评估系统运行状态,在必要的情况下发出迁移指令,调整各计算节点上 LP 分布. 基本算法框架如下.

```
Initial partitioning;
while GVT<SimEndTime do
{
  Collect performance information from host;
  Calculate computation workload imbalance WB;
  Calculate communication workload imbalance CB;
  if(WB>MaxLoadDiff) call balance_computation()
  else {
        if (CB>MaxCommDiff) call
          balance_communication()
      }
  if(MoveIndicator is not empty) Start LP movements;
  SetTheNextMeasurementTime;
}
```

4.1　初始化负载划分

初始化工作的主要目标是通过静态负载平衡的方法获得一个可行的 LP 初始分布,该分布是后续动态划分算法的基础:一个良好的初始分布能有效减少动态迁移所带来的开销. 由于仿真系统的事件调度依赖关系在运行前一般难以完全确定,所以负载划分的初始化工作主要考虑 LP 计算负载与计算节点容量的匹配.

计算节点的计算容量可以通过参考处理器时钟频率、内存大小等因素直接设置权重得到,也可以通过运行测试程序来获得更精确的初始计算容量参数 $Icap(i):N\rightarrow R$;得到所有计算节点基本参数后,计算 $dIcap(i)=\dfrac{Icap(i)}{\sum_{i\in N}Icap(i)}$ 以规范表示单个节点的初始计算容量,其中 $\sum_{i\in N}dIcap(i)=1$. 同时,用户可以根据 LP 的计算负载设置不同的初始权重 $ILoad(l):L\rightarrow R$,默认权重为 1,然后同样计算 $dILoad(l)=\dfrac{ILoad(l)}{\sum_{l\in L}ILoad(l)}$ 规范表示某个 LP 计算负载,其中 $\sum_{l\in L}dILoad(l)=1$. 最后根据 $dIcap(i)$ 和 $dILoad(l)$ 进行初始负载划分:

```
Initial Patitioning
Input: the LP collection L;
   the node collection N;
   the LP workload dILoad(l);
   the capacity of node dIcap(i);
Output: InitialPartition(i): N→L', L'⊆L
for (i∈N){
   let PartitionSet(i) be the LP collection which
   distributed to node i
   let SumLoad(i)= Σ_{l∈PartitionSet(i)} dILoad(l)
   While (SumLoad(i)<dIcap(i)){
      find l∈L satisfied SumLoad(i)+dILoad(l)<=
```

```
    dIcap(i)
    if(exit such LP l){
      PartitionSet(i):=PartitionSet(i)∪l
      SumLoad(i)=SumLoad(i)+dILoad(l)
      Remove l from L
     }else break;
  }
}
if(L≠∅) assign the rest LP to the N equally
InitialPartition(i)=PartitionSet(i)
```

4.2 运行信息采样与评估

跟踪系统运行信息并对系统状态进行评估是动态划分算法的基础，为了准确地了解整体系统的运行状态，需要对以下 3 个因素进行采样和评估：计算节点的计算容量 $cap(i)$、LP 计算负载 $LPload(l)$ 和 LP 的通信负载 $LPcomm(l)$.

计算容量表示计算节点在运行仿真时能提供的处理能力，容量越大的计算节点能在单位时间内进行的计算越多. 计算节点的基本容量由该节点的处理器、内存等硬件配置决定，但在仿真运行过程中，节点的计算容量会因为背景负载而改变，因此需要在运行时对节点计算容量进行采样和评估.

计算节点在采样时段内处理并提交的事件总数是衡量节点计算容量的一个重要指标[10-11]，但在乐观策略下回滚会导致部分实际处理了的事件无法被统计，此外，在一些受限乐观同步算法中（如 BTB、BSP），计算节点也会由于全局同步而产生空转等待，因此，单纯对处理并提交事件总数进行采用和评估无法准确地反映节点计算容量，需要考虑回滚和空转因素的影响. 与文献[16，19]的思想类似，我们设处理事件总数 $NodeEvent(i):N\rightarrow R$ 表示在采样时段内提交事件和回滚事件之和，实际处理时间 $NodeWorkTime(i):N\rightarrow walltime$ 表示采样区间时长减去空转等待时间，则定义节点 i 的动态计算容量为

$$cap(i)=\frac{NodeEvent(i)}{NodeWorkTime(i)}.$$

并行离散事件仿真中，单个 LP 的计算负载是通过处理该 LP 上的事件而产生的，因此计算在采样时段内该 LP 上处理的事件数是衡量计算负载的重要指标. 考虑到采样时段内该 LP 上处理的事件数实际会受节点计算能力的影响，分布到计算能力强的节点上的 LP 可能处理更多的事件，所以对 LP 计算负载的评估必须去除节点计算能力的影响. 我们采用计算 LP 推进单位虚拟时钟所产生的事件数来表示 LP 的计算负载，设 $LPEvent(l):L\rightarrow R$ 表示在采样时段内逻辑进程 l 上处理的事件数，$LVTAdvance:N\rightarrow virtualtime$ 表示在采样时段内逻辑进程 l 所在计算节点推进的局部仿真时钟，则定义逻辑进程 l 的计算负载为

$$LPLoad(l)=\frac{LPEvent(l)}{LVTAdvance}.$$

在乐观策略下，$LPEvent(l)$ 和 $LVTAdvance$ 的采样都会受到回滚的影响，为解决该问题，我们借鉴了 Schlagenhaft 等在文献[20]中提出的方法，对采样时段内的 l 上处理的所有事件记数，并在计算虚拟时钟推进时对回滚部分进行补偿.

单个 LP 的通信负载表示该 LP 调度其他 LP 上事件的情况，对 LP 通信负载的评估同样要去除节点计算能力的影响，设 $LPCommEvent(l_1,l_2):L\times L\rightarrow R$ 表示逻辑进程 l_1 在采样时段内调度逻辑进程上 l_2 的事件数，$LVTAdvance:N\rightarrow virtualtime$ 表示在采样时段内逻辑进程 l_1 所在计算节点推进的局部仿真时钟，则定义逻辑进程 l_1 的对逻辑进程 l_2 的通信负载为

$$LPcomm(l_1,l_2)=\frac{LPCommEvent(l_1,l_2)}{LVTAdvance}.$$

仿真平台中各节点上仿真控制器负责该节点上 LP 的事件调度处理和消息转发，因此，以上运行信息可方便地由各个计算节点上仿真控制器进行采样. 获取相关数据后，动态负载平衡算法则对整体运行状况进行评估.

计算 $dcap(i)=\dfrac{cap(i)}{\sum\limits_{i\in N}cap(i)}$ 以规范表示单个节点的计算容量，其中 $\sum\limits_{i\in N}dcap(i)=1$；计算 $dLPLoad(l)=\dfrac{LPLoad(l)}{\sum\limits_{l\in L}LPLoad(l)}$ 规范表示某个 LP 计算负载，其中 $\sum\limits_{l\in L}dLPLoad(l)=1$；设 $PartitionSet(i):N\rightarrow L$ 表示划分到节点 i 上的 LP 集合，计算 $NodeLoad(i)=\sum\limits_{l\in PartitionSet(i)}dLPLoad(l)$ 表示计算节点 i 上的计算负载，则计算负载均衡评估函数 WB 可以表示为 $WB=\mathrm{MAX}(|NodeLoad(i)-dcap(i)|)$，$\forall i\in N$，如果 $WB>MaxLoadDiff$ 则表示当前 LP 分布产生计算负载不平衡，需要进行计算负载平衡.

当 $l_1\in PartitionSet(i)\wedge l_2\in PartitionSet(i)$ 时表示 l_1，l_2 分布在同一计算节点上时，$LPcomm(l_1,l_2)$ 表示本地事件调度通信负载；否则，$LPcomm(l_1,l_2)$

表示远程事件调度通信负载.计算

$$NodeLocalComm(i)=\sum_{l_1\in PartitionSet(i)\wedge l_2\in PartitionSet(i)} LPcomm(l_1,l_2)$$

表示节点 i 上 LP 本地通信负载之和,则可得到系统本地通信负载之和 $W_l=\sum_{i=1}^{|N|}NodeLocalComm(i)$.计算

$$NodeComm(i,j)=\sum_{l_1\in PartitionSet(i)\wedge l_2\in PartitionSet(j)}(LPcomm(l_1,l_2)+LPcomm(l_2,l_1))$$

表示计算节点 i 和 j 间的 LP 通信负载总量,则进一步可以得到系统远程通信负载之和 $W_r=\sum_{i=1}^{|N|}\sum_{j=i+1}^{|N|}NodeComm(i,j)$.通信负载平衡评估函数 CB 可以表示为 $CB=\frac{W_r}{W_l}$,如果 $CB>MaxCommDiff$ 则当前 LP 分布产生通信负载不平衡,需要进行通信负载平衡.

4.3　计算负载平衡算法

计算负载平衡的基本目标是计算节点容量与分布到该节点上的 LP 计算负载的匹配以减少由各计算节点局部仿真时间(LVT)推进的差异而引起的额外开销,一个计算负载平衡的 LP 划分是进行后续通信负载优化算法的基础.算法的基本思想是在计算负载出现不平衡时,首先找到系统中计算负载最重的节点和计算负载最轻的节点:设 $Pmax$ 表示计算负载最重的节点,则 $Pmax$ 满足条件:

$$\forall i\in N,\ NodeLoad(Pmax)-dcap(Pmax)\geq NodeLoad(i)-dcap(i).$$

设 $Pmin$ 表示计算负载最轻的节点,则 $Pmin$ 满足条件:

$$\forall i\in N,\ NodeLoad(Pmin)-dcap(Pmin)\leq NodeLoad(i)-dcap(i).$$

将 $Pmax$ 上的部分 LP 迁移到 $Pmin$ 上,更新节点负载信息后重新评估计算负载是否平衡;如仍然存在计算负载不平衡,则重复上述步骤直到计算负载平衡;最后将整体迁移的最终结果存入迁移列表.

在选择迁移的 LP 时,为减少 LP 的迁移数量,则优先选择计算负载较大的 LP,具体算法如下:

```
balance_computation
Input: LP collection which distributed to node i
       PartitionSet(i)
  the LP workload dLPLoad(l)
  the capacity of node dcap(i)
Output: migrating LP collection MoveIndicator
while (WB>MaxLoadDiff) do{
  let Pmax be the node with the max computation
  workload
  let Pmin be the node with the min computation
  workload
  TMoveLoad=
      min(|NodeLoad(Pmax)-dcap(Pmax)|,
      |NodeLoad(Pmin)-dcap(Pmin)|)
  SumMoveLoad=0;
  For(l∈PartitionSet(Pmax)){
    If(SumMoveLoad+dLPLoad(l)≤TMoveLoad){
      Add l to TempMoveLPSet
      SumMoveLoad=SumMoveLoad+LPLoad(l);
      If(SumMoveLoad=TMoveLoad) break;
    }
  }
  NodeLoad(Pmax)=NodeLoad(Pmax)-
                 SumMoveLoad;
  NodeLoad(Pmin)=NodeLoad(Pmin)+
                 SumMoveLoad
  Add TempMoveLPSet to MoveIndicator
  if(TempMoveLPSet∩MoveIndicator≠∅) STOP;
  Calculate computation workload imbalance WB;
}
```

4.4　通信负载优化算法

在完成计算负载平衡迁移后,我们得到一个原始的 LP 划分,如该划分产生通信负载不平衡,则根据系统事件调度依赖关系对负载划分进行优化,减少由远程事件调度而引起的回滚和空转开销.根据第 3 节所述的基本问题模型,构造一个全连通的无向图 $G(L,E)$;采用 4.2 节所述的运行信息采样信息为无向图中边赋权重 $w:L\times L\to R$;$\forall l_1,l_2\in L$:$w(l_1,l_2)=LPcomm(l_1,l_2)+LPcomm(l_2,l_1)$,记该赋权图为 $G_w(L,E)$;则通信负载优化的目标即找到 $G_w(L,E)$的一个图划分,在保证各子图顶点个数符合计算负载平衡迁移结果的前提下,使得各子图间的边权重和最小.

设 $C_w=\sum w(l_1,l_2)$表示图 $G_w(L,E)$中所有边权重的总和,对任意一次通信负载优化,该值是一个常量.构造新的赋权图$\overline{G_w}(L,E)$,其中对边赋权重 $\overline{w}:L\times L\to R$;$\forall l_1,l_2\in L$:$\overline{w}(l_1,l_2)=C_w-w(l_1,l_2)$,则上述通信负载优化问题等价于找$\overline{G_w}(L,E)$的一个图划分,在保证各子图顶点个数符合计算负载平衡迁移结果的前提下,使得各子图间的边权重和最

大.该问题可基于 Gaur 等人在文献[21]中提出的图最大 k 划分算法获得近似解.

设 $\overline{w}(l_1,N_i)=\sum_{l_2\in N_i}\overline{w}(l_1,l_2)$ 表示图 $\overline{G_w}(L,E)$ 中逻辑进程 l_1 对节点 N_i 上所有逻辑进程的边权重和,$|N_i|$ 表示分布到节点 i 上 LP 的个数,则具体通信负载平衡算法如下:

```
balance_communication
Input: LP collection which distributed to node i
  PartitionSet(i)
  edge weight w(l1,l2) of interaction graph G_w(L,E)
  Accuracy ε>0
Output: migrating LP collection MoveIndicator
let i=0;
While(True){
  let W_r^i = Σw(l1,l2) while l1,l2 don't assign to the
  same node
  q_i = W_r^i ε / ((ε(|N|-1)+1)|L|), w'(l1,N_i) = ⌈w(l1,N_i)/q_i⌉ q_i
  do {
    find a pair of LP l1∈N_i, l2∈N_k satisfied
    |N_i|w'(l1,N_i)+|N_k|w'(l2,N_k)>
    |N_i|w'(l1,N_k)+|N_k|w'(l2,N_i)
      if(exit such pair){
        Move l1 to node N_k and l2 to node N_i
        Calculate W_new = Σw(l1,l2) while l1, l2
        don't assign to the same node
      }else{
       output the MoveIndicator;
       STOP;
      }
  }while (W_new>W_r^i/2)
 set i=i+1; W_r^i=W_new
}
```

性质 1. 算法收敛且最终分布与全局最优解比值不小于 $\left(1-\frac{1}{|N|}\right)(1-\varepsilon)$.

证明. 根据通信负载平衡算法迁移判断条件,则系统 LP 最终分布满足

$$\forall l_1\in N_i,l_2\in N_k$$
$$|N_i|\overline{w}'(l_1,N_i)+|N_k|\overline{w}'(l_2,N_k)\le(|N_i|\overline{w}'(l_1,N_k)+|N_k|\overline{w}'(l_2,N_i)) \tag{1}$$

则对所有 $l_1\in N_i$ 和 $l_2\in N_k$ 将上述不等式相加可以得到

$$|N_k||N_i|\sum_{l_1\in N_i}\overline{w}'(l_1,N_i)+|N_i||N_k|\sum_{l_2\in N_k}\overline{w}'(l_2,N_k)\le|N_k||N_i|\sum_{l_1\in N_i}\overline{w}'(l_1,N_k)+|N_i||N_k|\sum_{l_2\in N_k}\overline{w}'(l_2,N_i) \tag{2}$$

设
$$2\overline{w}'(N_i)=\sum_{l_1\in N_i}\overline{w}'(l_1,N_i),$$
$$2\overline{w}'(N_k)=\sum_{l_2\in N_k}\overline{w}'(l_2,N_k),$$
$$\overline{w}'(N_i,N_k)=\sum_{l_1\in N_i}\overline{w}'(l_1,N_k),$$
$$\overline{w}'(N_k,N_i)=\sum_{l_2\in N_k}\overline{w}'(l_2,N_i),$$

则不等式(2)可变化为

$$\overline{w}'(N_i)+\overline{w}'(N_k)\le\overline{w}'(N_i,N_k).$$

对系统中所有节点进行求和计算可以得到

$$(|N|-1)\sum_{i=1}^{|N|}\overline{w}'(N_i)+(|N|-1)\sum_{k=1}^{|N|}\overline{w}'(N_k)\le\sum_{i=1}^{|N|}\sum_{k=1,k\ne i}^{|N|}(\overline{w}'(N_i,N_k)).$$

设 $\overline{W_l'}=\sum_{i=1}^{|N|}\overline{w}'(N_i)$ 表示 $\overline{G_w}(L,E)$ 各子图内部边权重和,$\overline{W_r'}=\sum_{i=1}^{|N|}\sum_{k>i}^{|N|}(\overline{w}'(N_i,N_k))$ 表示 $\overline{G_w}(L,E)$ 各子图间边权重和,则可得到

$$(|N|-1)\overline{W_l'}\le\overline{W_r'} \tag{3}$$

设 $\overline{W'}$ 和 q' 表示算法中外层循环运行 $\overline{W_r^i}$ 和 q_i 的最终值,可得 $\overline{W_r'}\ge\overline{W'}/2$,则

$$\begin{aligned}\overline{W_r'}&=\sum_{i=1}^{|N|}\sum_{k>i}^{|N|}\Big(\sum_{l_1\in N_i}\overline{w}'(l_1,N_k)\Big)\\&\le\sum_{i=1}^{|N|}\sum_{k>i}^{|N|}\Big(\sum_{l_1\in N_i}\Big(\frac{\overline{w}(l_1,N_k)}{q'}+1\Big)q'\Big)\\&=\sum_{i=1}^{|N|}\sum_{k>i}^{|N|}\Big(\sum_{l_1\in N_i}\overline{w}(l_1,N_k)\Big)+\Big(\frac{|N||L|}{2}\Big)q'\\&=\overline{W_r}+\frac{|N||L|}{2}\cdot\frac{\overline{W'}\varepsilon}{(\varepsilon(|N|-1)+1)|L|}\\&\le\overline{W_r}+\frac{|N||L|}{2}\cdot\frac{2\,\overline{W_r'}\varepsilon}{(\varepsilon(|N|-1)+1)|L|}\\&=\overline{W_r}+\frac{\overline{W_r'}|N|\varepsilon}{\varepsilon(|N|-1)+1},\end{aligned}$$

即 $\overline{W_r'}\le\frac{(|N|-1)\varepsilon+1}{1-\varepsilon}\overline{W_r}$.

根据不等式(3)可以得到

$$(|N|-1)\overline{W_l'}\le\overline{W_r'}\le\frac{(|N|-1)\varepsilon+1}{1-\varepsilon}\overline{W_r},$$

则 $\overline{W_l}\le\overline{W_l'}\le\frac{\varepsilon(|N|-1)+1}{(|N|-1)(1-\varepsilon)}\overline{W_r}$.

全局最优解表示图 $\overline{G_w}(L,E)$ 划分后子图间边权重和的最大值,设为 $\overline{W_r^{opt}}$,则

$$\overline{W_r^{opt}} \le \overline{W_r} + \overline{W_l} \le \overline{Wr} + \frac{\varepsilon(|N|-1)+1}{(|N|-1)(1-\varepsilon)}\overline{Wr},$$

所以 $\dfrac{\overline{Wr}}{\overline{W_l^{opt}}} \ge \left(1-\dfrac{1}{|N|}\right)(1-\varepsilon)$.

性质 2.　算法具有多项式时间复杂度,且时间复杂度不超过 $O(n^4\varepsilon^{-1}\log n)$.

证明.　设 $n=|L|$,因为对边权重进行了取整缩放 $\overline{w}'(l,N)=|\overline{w}(l,N)/q|q$,所以内层 while 循环执行时间复杂度是 $O(n(1+\varepsilon)/\varepsilon)=O(n/\varepsilon)$. 为证明外层 while 循环时间复杂性,需要首先引入 Radzik 等人在文献[22]中证明的定理.

引理.　设 $\boldsymbol{d}=(d_1,\cdots,d_n)$ 表示一个向量,$\boldsymbol{y}_1,\cdots,\boldsymbol{y}_n$ 表示一组 $\{0,1\}^n$ 向量,如果对所有 $i=1,\cdots,p-1$,都满足 $0\le \mathrm{d}y_{i+1}\le 1/2\mathrm{d}y_i$,则 $p=O(n\log n)$.

由于每次外层循环 $\overline{W_r^{i+1}}\le\dfrac{1}{2}\overline{W_r^i}$,且 $\overline{W_r}$ 可以看作是所有边权重与 $\{0,1\}^n$ 向量的内积,其中远程调度边对应 1,本地调度边对应 0,所以根据引理外层 while 循环时间复杂性不会超过 $O(n\log n)$. 又因为每次搜索符合条件的点不超过 $O(n(n-1)/2)=O(n^2)$,所以算法的时间复杂性不超过 $O(n^4\varepsilon^{-1}\log n)$.

5　实验分析

5.1　实验环境

本文实验基于并行离散事件仿真领域经典的 PHOLD 测试模型[23],同时修改其事件调度方式以表示更复杂的模型交互模式:实验仿真系统由一系列实体构成,每个实体映射到相应 LP 上,负责处理相关事件,每个事件随机做若干次加法以表示不同计算负载;当某个实体处理事件时,该实体随机调度其他实体上的事件,设被处理事件的逻辑时戳为 t,则随机选取区间 $[t+1,t+dmax]$ 中任意时刻作为新调度事件的时戳;同时,仿真系统中所有实体分成若干不相交的子集,某个实体调度同组实体上事件的概率为 $Pgroup$,调度不同组实体上事件的概率为 $1-Pgroup$. 实际测试过程中,实体总数设为 1000 个,每个实体初始事件数为 2,$dmax=10$,$Pgroup=0.8$,整个仿真运行的逻辑时间为 20000. 实验仿真软件平台为国防科学技术大学计算机学院研制的 YH-SUPE 并行仿真开发及运行支撑环境[24]. 实验硬件环境配置为 8 计算节点的集群系统,每个计算节点 CPU 主频 3.0GHz,主存 1GB,操作系统为红帽 Linux.

5.2　实验结果

实验主要关注两个方面的:整体仿真性能加速比和 LP 迁移收敛情况,主要对比测试对象为德国波恩大学 Peschlow 于 2007 年发表的 DynPart 算法[16].

基于以上改进的 PHOLD 测试模型,本次实验中我们采用 3 种交互配置,分别将所有实体分为 50 组、25 组和 1 组. 由图 1 测试结果可以看到 DynPart 算法在模型交互模式没有明显特征的情况下(group1)其迁移较为盲目,一直保持较高的额外的开销;而在交互模式特征较为明显的情况下(25 组,50 组)算法迁移情况收敛较快. 本文 ALSPart 算法则一直保持较低的迁移开销.

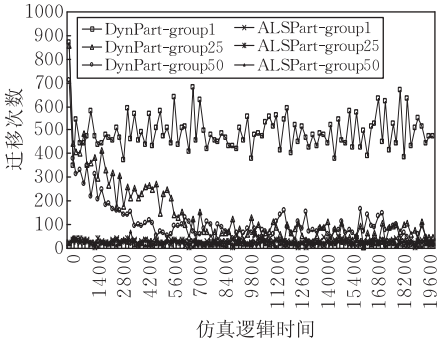

图 1　实体迁移次数随仿真时间变化情况

图 2～图 4 分别显示了在时间弯曲(Time Warp)、呼吸时间桶(Breathing Time Bucket)和呼吸时间弯曲(Breathing Time Warp) 3 种乐观同步协议下仿真运行时间;图 5～图 7 分别显示了相应同步协议下仿真回滚次数. 由测试结果可以看到采用本文 ALSPart 算法平均减少 38.11% 的回滚次数,而且整体性能较为稳定,通过优化通信负载对 3 种交互配置分别可以降低 13.64%、49.63% 和 51.05% 的回滚次数,同时对 3 种不同的同步协议则分别可以减少 44.09%、29.65% 和 40.59% 的回滚次数;而 DynPart 算法性能对交互配置较为敏感,特别是在交互配置 1 组的情况下会由于不适当的划分而产生大量额外的回滚,同时 DynPart 在呼吸时间桶同步协议下整体性能较差,此外,在实际测试中,DynPart 的整体性能非常依赖对其运行参数的配置,不适合的参数配置将导致其性能进一步下降. DynPart 的优势在于在模型交互模式特征较为明显的情况下(25 组,50 组)其性能较好. 需要说明的是本次实验中动态负载划分算法对运行时间的加速比

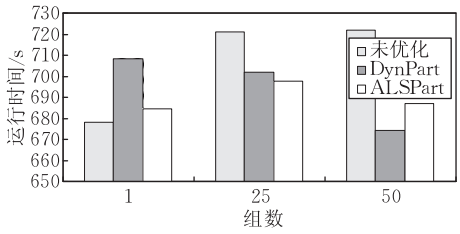

图 2 时间弯曲同步协议下仿真运行时间

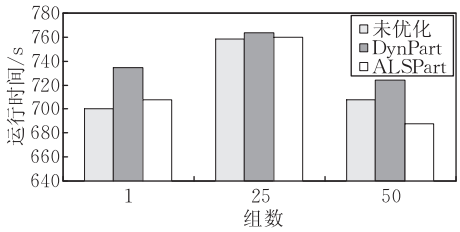

图 3 呼吸时间桶同步协议下仿真运行时间

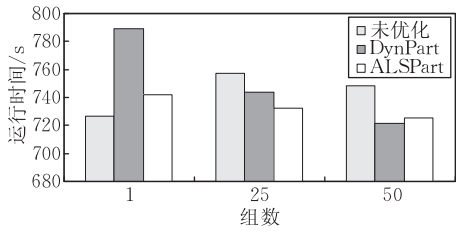

图 4 呼吸时间弯曲同步协议下仿真运行时间

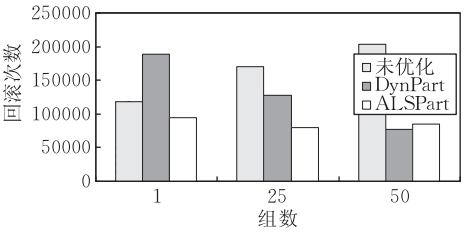

图 5 时间弯曲同步协议下事件回滚次数

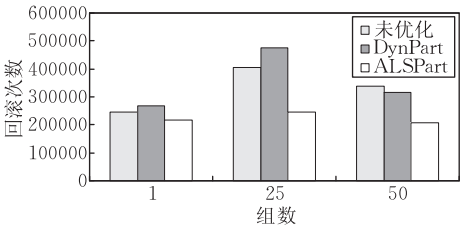

图 6 呼吸时间桶同步协议下事件回滚次数

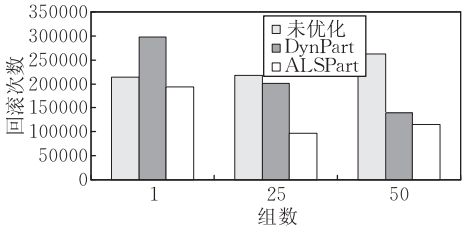

图 7 呼吸时间弯曲同步协议下时间回滚次数

没有对回滚次数的加速比明显，这主要是由于本次 PHOLD 测试程序中每个事件的计算负载较轻，使得回滚的开销不大，因此整体运行时间差别不明显；而在实际应用中回滚将极大地影响仿真运行性能.

6 结 论

负载划分是决定并行离散事件仿真性能的重要因素，如何度量和根据模型交互模式优化通信负载是目前动态负载划分研究的热点. 本文将并行离散事件仿真负载划分归结为一个带权重无向图有限容量 k 划分问题，提出了一个多项式时间复杂度的基于顶点交换的启发式局部搜索近似划分算法，旨在使计算负载平衡的前提下通信负载最优化. 理论证明该算法近似解与全局最优解比值不小于 $(1-1/|N|)(1-\varepsilon)$，时间复杂度不超过 $O(n^4\varepsilon^{-1}\log n)$. 测试结果表明该算法平均减少 38.11% 的回滚开销，能有效地提高大规模并行离散事件仿真的运行性能. 在下一步工作中，我们将结合 multiway cut 问题模型对现有的划分模型进行扩展，引入对特定 LP 节点的优化处理，以解决现有算法在特殊计算负载分布下效率较低的问题，进一步提升仿真运行性能.

参 考 文 献

[1] Fujimoto R M. Parallel and Distributed Simulation Systems. New York: John Wiley & Sons Inc, 2000

[2] Chen D, Szymanski K B. Dsim: Scaling time warp to 1033 processors//Proceedings of the 2005 Winter Simulation Conference. Orlando, FL, USA, 2005: 346-355

[3] Lemeire J, Smets B, Cara P, Dirkx E. Exploiting symmetry for partitioning models in parallel discrete event simulation//Proceedings of the 18th Workshop on Parallel and Distributed Simulation (PADS'04). Kustein, Austria, 2004: 189-194

[4] Boukerche A, Fabbri A. Partitioning parallel simulation of wireless networks//Proceedings of the 2000 Winter Simulation Conference. Orlando, FL, USA, 2000: 1449-1457

[5] Vulov G, He Tianhao, Hybinette M. Quantitive assessment of an agent-based simulation on a time warp executive//Proceedings of the 2008 Winter Simulation Conference. Miami, FL, USA, 2008: 1068-1076

[6] Pawlaszczyk D, Timm I J. A hybrid time management approach to agent-based simulation//Proceedings of the 29th Annual German Conference on Artificial Intelligence (KI 2006). Bremen, Germany, 2006: 374-388

[7] MOOSE-Module-based Object-Oriented Simulation Environment. http://web.cs.uni-bonn.de/IV/MOOSE/, 2007

[8] Barr R, Hass Z J, Renesse R. Jist: An efficient approach to simulation using virtual machines. Software-Practice and Experience (SPE), 2005, 35(6): 539-576

[9] Shen W. Load migration policies in distributed simulation using SPEEDES [M. S. dissertation]. New Hampshire: Dartmouth College, Hanover, 1998

[10] Reiher P L, Jefferson D. Virtual time based dynamic load management in the time warp operating system//Proceedings of the 1990 SCS Multiconference on Distributed Simulation. San Diego, CA, USA, 1990: 103-111

[11] Glazer D W, Tropper C. On process migration and load balancing in time warp. IEEE Transactions on Parallel and Distributed Systems, 1993, 4(3): 318-327

[12] Carothers C D, Fujimoto R M. Efficient execution of time warp programs on heterogeneous NOW platforms. IEEE Transactions on Parallel and Distributed Systems, 2000, 11(3): 299-317

[13] Wilson L F, Shen W. Experiments in load migration and dynamic load balancing in SPEEDES//Proceedings of the 1998 Winter Simulation Conference. Washington DC, USA, 1998: 483-490

[14] Slavik M, Mahgoub I, Badi A. Dynamic entity distribution in parallel discrete event simulation//Proceedings of the 2008 Winter Simulation Conference. Miami, FL, USA, 2008: 1061-1067

[15] Low M Y H. Dynamic load-balancing for BSP time warp//Proceedings of the 35th Annual Simulation Symposium (SS'02). San Diego, CA, USA, 2002: 267-274

[16] Peschlow P, Honecker T, Martini P. A flexible dynamic partitioning algorithm for optimistic distributed simulation//Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation (PADS'07). San Diego, CA, USA, 2007: 219-228

[17] Even G, Naor J, Rao S, Schieber B. Fast approximate graph partitioning algorithms. SIAM Journal on Computing, 1999, 28(6): 2187-2214

[18] Nagarajan V, Ravi R. Approximation algorithms for requirement cut on graphs. Algorithm. New York: Springer, 2008

[19] Reiher P L, Jefferson D. Virtual time based dynamic load management in the time warp operating system//Proceedings of the SCS Multiconference on Distributed Simulation. San Diego, CA, USA, 1990: 103-111

[20] Schlagenhaft R, Ruhwandl M, Sporrer C, Bauer H. Dynamic load balancing of a multi-cluster simulator on a network of workstations//Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS'95). Los Alamitos, CA, USA, 1995: 175-180

[21] Gaur D R, Krishnamurti R, Kohli R. The capacitated max-k-cut problem//Proceedings of the International Conference on Computational Science and Its Applications (ICCSA). Singapore, 2005: 670-679

[22] Radzik T. Parametric flows, weighted means of cuts, and fractional combinatorial optimization//Proceedings of the Complexity in Numerical Optimization, World Scientific, 1993: 351-386

[23] Fujimoto R M. Performance of time warp under synthetic workload//Proceedings of the SCS Multiconference on Distributed Simulation. San Diego, CA, USA, 1990: 23-28

[24] Yao Yi-Ping, Zhang Ying-Xing. A solution for analytic simulation based on parallel processing. Journal of System Simulation, 2008, 20(24): 6617-6621(in Chinese)
(姚益平, 张颖星. 基于并行处理的分析仿真解决方案. 系统仿真学报, 2008, 20(24): 6617-6621)

ZHANG Ying-Xing, born in 1981, Ph. D.. His current research interests focus on parallel discrete event simulation and agent-based simulation.

YAO Yi-Ping, born in 1963, professor, Ph. D. supervisor. His major research interests include parallel and distributed simulation and virtual reality.

Background

Discrete-event simulation (DES) provides a powerful technique for modeling the intricate interactions of complex systems. Over the last decades, many researchers aim at overcoming the performance bottleneck of a sequential discrete-event simulation (SDES) by distributing the DES across multiple processing elements in parallel discrete event simulation (PDES) system. The partitioning mechanism is regarded as one of the most important technical issue which seriously influences the performance of PDES as well as synchronization protocol. Since both the model behavior and the amount of available resources might change frequently throughout the whole simulation, the dynamic partitioning algorithm is required to gain better speedup. Currently, there have been developed several dynamic partitioning algorithms, however, most of them do not pay enough attention to communication structure and only consider computation load imbalances. This paper formalizes the dynamic partitioning problem statement which combine both computation and communication load balancing, and proposes a algorithm based on an approximate algorithm for graph partitioning.

This work is supported by the National Natural Science Foundation of China under grant No. 60773019 and the Ph. D. Programs Foundation of Ministry of Education of China under grant No. 200899980004. The aim of the research is to develop algorithms and techniques for PDES simulation kernel which supports complex application, such as agent-based simulation. Until now, the research team has published more than 20 papers about the technology of parallel and distributed simulation.