Journal of System Simulation

Volume 30 | Issue 6 Article 26

6-14-2018

Parallel Design and Load Balance of OpenMP Based Warfighting Simulation

Wang Xiao
1. Electronic Engineering Institute, Hefei 230037, China;;

Yaqi Liu
1. Electronic Engineering Institute, Hefei 230037, China;;

Yuben Tao
2. Unit 31102 of the PLA, Nanjing 210000, China;

Parallel Design and Load Balance of OpenMP Based Warfighting Simulation

Abstract

Abstract: Parallel simulation is a main trend of large scale and high precision war fighting simulation (WS). However, currently there is no unified framework to design a parallel WS and no effective method to load balance of it. *Based on the parallel agent theory, an OpenMP based parallel WS design framework (PWSF) is proposed. To solve the problem of the complex simulation load distribution resulted from the diversity of entities' models and the position independence of the interactions, a load balance method based on METIS multilevel graph partition tool is suggested.* Based on the improved PHOLD test model, experiments on different types and amounts of simulation loads and different numbers of CPU cores are conducted. The results show that METIS based PWSF is with good universality, timeliness and expandability.

Keywords

OpenMP, parallel war fighting simulation (PWS), METIS, load balance

第 30 卷第 6 期 系统仿真学报© Vol. 30 No. 6
2018 年 6 月 Journal of System Simulation Jun., 2018

基于 OpenMP 的作战仿真并行设计与负载均衡研究

王枭[1]，刘雅奇[1]，陶玉犇[2]
(1. 电子工程学院，安徽 合肥 230037；2. 中国人民解放军 31102 部队，江苏 南京 210000)

摘要：并行仿真是大规模、高精度作战仿真的主流趋势，但目前尚缺乏统一的设计机制和有效的负载均衡方法。对此，*借鉴并行 Agent 理论，提出了基于 Open MP 的作战仿真并行设计框架(PWSF)。针对由实体模型的多样性及其交互的位置无关性引起的复杂负载分布，提出利用 METIS 多级图划分工具进行负载均衡的方法。*在改进的 PHOLD 测试模型上，分别对不同类型和数量级的仿真负载，以及不同 CPU 核心数的情况进行了实验，结果显示，基于 METIS 负载均衡的 PWSF 具有良好的普适性、高效性和可扩展性。

关键词：OpenMP；并行作战仿真；METIS；负载均衡
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2018) 06-2206-10
DOI: 10.16182/j.issn1004731x.joss.201806026

Parallel Design and Load Balance of OpenMP Based Warfighting Simulation

Wang Xiao[1], *Liu Yaqi*[1], *Tao Yuben*[2]
(1. Electronic Engineering Institute, Hefei 230037, China; 2. Unit 31102 of the PLA, Nanjing 210000, China)

Abstract: Parallel simulation is a main trend of large scale and high precision war fighting simulation (WS). However, currently there is no unified framework to design a parallel WS and no effective method to load balance of it. *Based on the parallel agent theory, an OpenMP based parallel WS design framework (PWSF) is proposed. To solve the problem of the complex simulation load distribution resulted from the diversity of entities' models and the position independence of the interactions, a load balance method based on METIS multilevel graph partition tool is suggested.* Based on the improved PHOLD test model, experiments on different types and amounts of simulation loads and different numbers of CPU cores are conducted. The results show that METIS based PWSF is with good universality, timeliness and expandability.

Keywords: OpenMP; parallel war fighting simulation (PWS); METIS; load balance

引言

作战仿真是军事研究的重要手段。当前，越来越多的作战实体被加入仿真想定，实体模型也越来越逼真、细致，作战仿真正朝着大规模、高精度方向发展，由此带来的计算量问题成为了作战仿真面临的一项主要困难。随着摩尔定律逐渐失效，处理器性能发展趋于极限，并行计算技术逐渐成为提高仿真效率的主要手段。在作战仿真领域，并行计算已有不少成功的应用，如文献[1]给出了分导式多弹头导弹弹道的并行仿真方法，文献[2]研究了协同空战火力分配的并行仿真方法，等。这些并行方法多植根于具体的专业领域特征，能够有效解决高

收稿日期：2016-08-09 修回日期：2016-11-10；
基金项目：全军军事类研究生资助课题(2013JY312)；
作者简介：王枭(1991-)，男，安徽寿县，博士生，研究方向为军事仿真与现代智能算法；刘雅奇(1965-)，男，陕西岐山，博士，教授，博导，研究方向为电子对抗作战效能分析，作战模拟与作战实验。

精度仿真问题，但却难以拓展适配于其他军兵种专业。大规模作战仿真的并行计算问题还没有得到有效解决。究其原因，主要在于：一、开放性的仿真模型缺乏统一的并行机制；二、交互关系错综复杂，负载难以均衡。

近年来，Multi-Agent System(多智能体系统)为大规模群体仿真提供了相对统一的建模范式。Agent 是具有自治、独立、交互等特性的行为单元[3]，基于 Agent 的作战实体建模为作战仿真提供了内在的并行性。当前，作战实体广泛采用组件化建模思路，计算任务被分散到传感器、行为决策、通信设备、火力单元等诸多可重用的模型组件中[4-5]。虽然这些模型组件的结构设计各不相同，但它们总体的仿真流程仍遵循“sense-think-act”的 Agent 主流计算范式[6-8]，这为统一的作战仿真并行设计框架奠定了基础。

负载均衡是并行 Agent 仿真(Parallel Agent Simulation，PAS)的重难点问题[9]，其总体要求是将并行仿真任务按一定规则分配到各逻辑线程，使总的计算时间趋于最小[10]。聚类是 PAS 负载均衡的常见方法[11-12]，但其默认的前提条件是交互仅在一定距离范围(Area of Interest)内的 Agent 之间发生，但实际作战中实体间的大量交互是通过 C4ISR 系统发生而并不受限于地理空间，因此对并行作战仿真(Parallel War fighting Simulation，PWS)的负载均衡需要寻找另外的针对性方法。

针对以上问题，本文在共享存储并行编程规范 Open MP 的基础上，对通用的作战仿真并行设计框架及其负载均衡方法展开研究。

1 基于 Open MP 的作战仿真并行设计框架

1.1 Open MP 概述

OpenMP[13-14]是一种共享存储的并行编程规范，由一系列环境变量、编译制导指令及函数库组成。其并行任务间通过共享内存实现通信，因此非常适用于当前的多处理器或多核心处理器计算机。其并行执行模型为 Fork-Join 模式：程序开始时，主线程串行执行；遇到并行宏指令后，派生出一组线程共同完成并行任务；任务完成后，派生线程挂起或退出，主线程继续执行串行程序，直到遇到下一个派生点。

OpenMP 提供了 3 种内置的任务调度策略：static、dynamic 和 guided。其中 static 一次性地将并行任务按数量平均分配到各线程；dynamic 依次将各并行任务实时地分配到空闲线程；guided 介于 static 和 dynamic 之间，依次递减地将一定数量的并行任务实时分配给各线程。这些调度策略的适用前提是各并行任务之间没有相互依赖关系，而作战仿真实体间交互频繁，因此 PWS 首先需要对仿真实体做去相关处理。

运行在不同处理器上的并行任务可通过共享内存实现数据交换。为了避免同时操作共享变量时发生访问错误，需要在写入数据时采用锁、原子或临界区等保护措施，这导致了 Open MP 并行仿真相对串行仿真的额外通信开销。如何对仿真任务做出合理地划分，使并行加速性能提高的同时，额外的任务调度与通信开销尽量减小，是 PWS 另一个需要重点关注的问题。

1.2 作战仿真并行设计框架

根据 Agent 建模理论[8]，作战实体的仿真过程可抽象为 sense-think-act 范式，即在每个仿真步长内，主动地从环境感知、获取信息，并根据这些信息及自身的行为规则、知识体系进行判断、决策，然后产生改变自身、环境甚至其他实体状态的行动。基于这一统一的建模规范，作战仿真任务在每一步长都可轻易地被划分到若干并行线程，各线程内串行执行若干实体的 sense-think-act 程序。图 1 给出了基于 PAS 理论的作战仿真并行设计框架(PWS Framework，PWSF)。

其仿真流程如下：

Step 1：仿真开始时，首先从数据库读入想定

第 30 卷第 6 期 系统仿真学报 Vol. 30 No. 6
2018 年 6 月 Journal of System Simulation Jun., 2018

实体链表，初始化交互链表。

Step 2：新的仿真步长开始时，根据前一步长的负载统计数据计算下一步长的负载分配方案，并通过 first private 语句为各线程拷贝私有的实体链表。

Step 3：各线程按照既定的负载分配方案，分别串行仿真分配到的若干实体。各实体的仿真流程如下：

(1) 读取历史交互链表，计算交互结果，更新实体状态。对应于 Agent 的 sense 操作。

(2) 检索实体任务链表，并根据当前状态，决策本步长的行为任务。对应 Agent 的 think 操作。

(3) 根据当前任务，执行行为仿真，更新本地私有实体链表，记录实体仿真时间。对应 Agent 的 act 操作。

(4) 记录仿真产生的交互次数与交互对象，并将跨线程交互写入当前步长交互链表。

Step 4：各线程计算完成后，将各线程的私有实体链表规约到全局链表中，同时将历史交互链表清空，再将当前交互链表交接到历史交互链表。

Step 5：如果未到仿真结束时间，则回到 Step 2；否则，输出仿真结果，结束仿真。

需要特别说明的是，为减少通信和便于进一步的负载均衡，该 PWSF 中融入了一些针对性的设计，具体是：

(1) 私有的实体链表

如前所述，写共享内存操作需要额外的通信开销，而实体在交互和行为仿真阶段有大量零散的状态更新需要写共享内存。PWSF 为每个子线程拷贝了私有的实体链表，本地实体的状态更新可通过私有实体链表，在线程仿真完成后做统一规约，这样可大量减少中间状态更新带来的通信。

(2) 动态的交互链表

实体行为仿真的结果包括对自身和其他实体状态的更新。当目标实体的仿真任务在本线程时，可直接通过私有实体链表对其进行更新；否则，需要将更新内容存入全局的交互链表，并在目标实体下一步长的交互仿真中执行。为精简交互链表的内容，PWSF 提供了“历史”和“当前”两个动态的交互链表，并在每一步长仿真结束时用“当前”链表覆盖“历史”链表，同时清空当前链表。

(3) 负载分布信息采集器

为便于定量均衡负载，PWSF 在每个实体的仿真过程中插入计时器和计数器，记录实体的仿真耗时、交互次数和交互对象(包括线程内和跨线程)，用于表征计算量和通信量的分布。通常，作战仿真的想定态势并非一成不变，实体的负载分布往往也随着时间推进有所变化。但这种变化一般不是无序的剧变，而是连续的渐变。因此，将前一步长的负载统计信息作为本步长负载分配的依据在多数情况下是合理的。

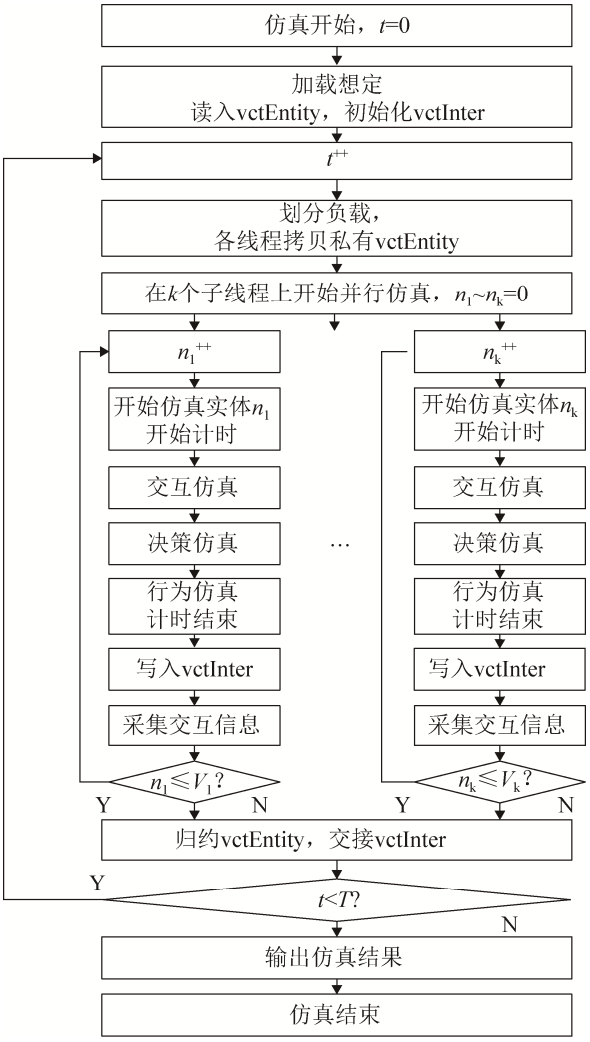


图 1 作战仿真并行设计框架

Fig. 1 Parellel Design Framework of Warfight Simulation

1.3 PWSF 基准测试模型

为了验证 PWSF 性能及对各种负载均衡策略进行综合测试，需要为 PWSF 建立基准测试模型。文献[15]在经典的 PHOLD 模型基础上，给出了 PAS 的合成基准测试模型。而 PWS 与理论的 PAS 相比，还具有模型开放性的特点，即作战仿真具体的实体模型由用户定制。也就是说各 Agent 的仿真计算量、交互次数和交互对象都各不相同，甚至 Agent 的数量都会因想定而不同。对此，在 PAS 基准测试模型的基础上，本文提出如下 PWSF 的基准测试模型：

```
1 //仿真开始，随机初始化 N 个实体的仿真负载
2 for(int n=0;n<N;n++){
3       vctEntity[n].Init();}
4 clock_t t1=clock();
5 //时间步长推进
6 for(int t=0;t<T;t+=detaT){
7       //划分负载：将 N 个实体分为 k 组
8      Load Balance(vctEntity, vctLoad, k);
9      //拷贝私有 vctEntity
10      vctEntityPriv=vctEntity; vctEntity.clear();
11      //开启 k 个线程，开始并行计算
12#pragma omp parallel for schedule(static) firstprivate
(vctEntityPriv)
13      for(int i=0;i<m;i++){
14          for(int j=0;j<vctLoad[i].size();j++){
15                                                  CEntity&
entity=vctEntityPriv[vctLoad[i][j] ];
16            //实体仿真，统计实体仿真耗时
17            clock_t t2=clock();entity.nData=0;
18            for(int m=0;m<entity.nCalNum;m++)
19                {entity.nData+=1;}
20            entity.nSimTime=clock()-t2;
21            //写跨线程交互
22            omp_set_lock(&ompLock);
23            for(int m=0;m<entity.nInterNum;m++){
24
if(!IsInVct(entity.vctInterTgt[m],vctLoad[i])){
25                CInteraction inter;
26                inter.nTgt=entity.vctInterTgt[m];
27                                    inter.nDataID=rand();
inter.nData=rand();
28                vctInter.push_back(inter);} }
29            omp_unset_lock(&ompLock);}
30       //归约 vctEntity
31       omp_set_lock(&ompLock);
32       for(int j=0; j<vctLoad[i].size(); j++){
33                    vctEntity.push_back(vct      Entity
Priv[vctLoad[i][j]]); } }
34           omp_unset_lock(&ompLock);}
35      //交接 vctInter
36      vctInterPre=vctInter; vctInter.clear();}
37//输出仿真总耗时
    38printf("Total Time=%d", clock()-t1);
```

其中，2~3 行在初始化阶段对各实体赋予指定分布的仿真计算量 nCalNum、交互次数 nIterNum 和交互对象 vctInterTgt，以模拟作战仿真模型的开放性；第 8 行执行负载均衡算法，将 N 个仿真实体按照核心数分为 k 组，并将分组方案存入 vct Load；第 12 行执行 OpenMP 的 static 调度，将 k 组负载分配给 k 个线程，并通过 first private 子句为各线程拷贝私有的实体链表；17~20 行模拟实体的仿真计算过程，并统计仿真耗时；22~29 行模拟实体的跨线程交互，并统计交互分布信息；31~34 行执行实体链表的规约；36 行执行交互链表的交接；38 行输出仿真总耗时。表 1 给出了提供给用户的测试接口与说明：

表 1 PWSF 基准测试模型接口与说明
Tab. 1 Interfaces and Descriptions of PWSF Benchmark Model

接口	说明
int k;	处理器核心数
int N;	仿真实体数
void CEntity::Init();	按照指定分布，初始化测试负载
void Load Balance(vector<CEntity>&, vector<vector<int> >&, int)	负载均衡算法

1.4 实验分析

为了对 PWSF 的并行性能有一个直观的了解，我们以实体数量均分的负载均衡策略为例，对大(1e7~1e8)、中(1e6~1e7)、小(1e5~1e6)计算量的 3 种分布(分布函数如表 2 所示)，大(20~30)、中(10~20)、小(0~10)交互次数的 3 种分布(分布函数同表 2)，共 9 种负载分布类型进行实验。

https://dc-china-simulation.researchcommons.org/journal/vol30/iss6/26
DOI: 10.16182/j.issn1004731x.joss.201806026

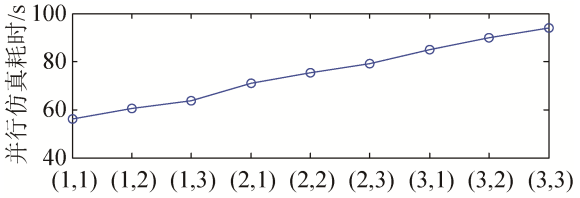
第 30 卷第 6 期 系统仿真学报 Vol. 30 No. 6
2018 年 6 月 Journal of System Simulation Jun., 2018

表 2 典型实体负载分布类型
Tab. 2 Typical Entity Load Distribution Patterns

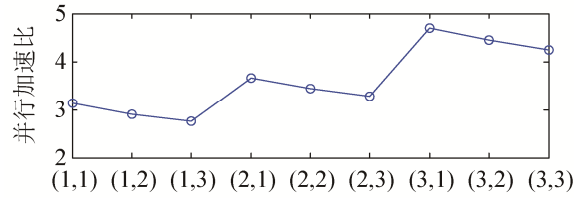
分布类型	分布概率		
	大	中	小
1	0.2	0.3	0.5
2	0.3	0.4	0.3
3	0.5	0.3	0.2

实验环境为：AMD FX-8 300 八核处理器(3.3 GHz)，8.0G 内存，Windows 7 SP1 操作系统。PWSF 的默认参数为：m=8，N=1 000，仿真 10 个步长。

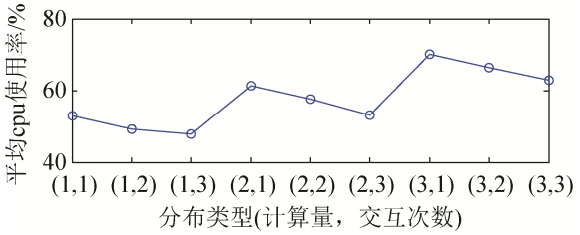
实验结果如图 2 所示。



(a) 仿真耗时



(b) 加速比



(c) 平均 CPU 使用率

图 2 典型负载分布情况下的 PWSF 性能测试
Fig. 2 Perfromance Test of PWSF under Typical Load Distribution Patterns

(1) 根据图 2(a)，并行仿真耗时随实体计算量和交互次数的增加而增加，且受交互次数的影响更为显著：交互次数增加 10 与计算量增加 10^8 带来的并行仿真耗时增加大约在一个数量级。

(2) 根据图 2(b)，并行加速比随着实体计算量增加而增加，随着交互次数的增加而减少。这主要是因为：首先，PWSF 在每个步长都需要一系列串行的负载均衡和数据同步(包括规约实体链表、交接交互链表等操作)操作，而这些操作的计算量是相对固定的，如果并行计算量(即实体计算量)较小，则 CPU 将频繁在串行和并行之间切换，限制平均 CPU 使用率的提升；其次，跨线程的交互需要通过以独占形式写共享内存实现，这一过程实际上也是串行的，因此交互次数的增加，也将提高 CPU 工作模式的切换频率，限制并行性能的提升。这一特性，在图 2(b)、(c)得到了印证。

(3) 根据图 2(b)，PWSF 对这九种测试负载的并行加速比都在 3~5 左右，虽然实现了一定程度的加速，但还远未达到理想的 8 倍加速。这主要是因为：数量均分策略多数情况下并不能对不规则分布的仿真负载做出高效的划分。PWSF 的负载均衡还需要更深入的研究。

2 基于 METIS 的 PWSF 负载均衡

2.1 PWSF 负载均衡模型

分析图 1 所示的作战仿真并行设计框架，并行仿真总耗时可表示为：

$$T_{PWS}=\sum_{t=1}^{T}\left[T_{LB}+\max_{i=1,2,\cdots,m}\left(\sum_{n=1}^{N_m}T_s(i,n)\right)+\sum_{n}^{N}T_i(n)+T_{SN}\right] \tag{1}$$

式中：T_{LB} 为负载均衡耗时；$T_s(i,n)$ 为线程 i 上实体 n 的仿真计算耗时；$T_i(n)$ 为实体 n 的跨线程交互耗时；T_{SN} 为数据同步耗时；T 为仿真的总步长数。

理论上，负载均衡(Load Balance)的目标是实现计算量平均化和通信开销最小化。注意到(1)式中，T_{SN} 几乎不受负载均衡策略影响，可视为常数，因此 PWSF 的负载均衡可表示为如下规划模型：

$$\begin{aligned}&\min T_{LD}=\max_{i=1,2,\cdots,m}\left(\sum_{n=1}^{N_m}T_s(i,n)\right)+\sum_{n}^{N}T_i(n)\\&s.t.\ P(n)\in[1,2,\cdots,m]\ \ n=1,2,\cdots,N\end{aligned} \tag{2}$$

式中：$P(n)$ 表示实体 n 被分配到的线程。

基于作战仿真连续渐变的假设，$T_s(i,n)$ 可直接

通过 PWSF 从前一步长仿真中采集，而 $\sum_{n}^{N} T_i(n)$ 正比于前一步长跨线程交互的总数 $\sum_{n=1}^{N} num\{I(n)|P(n)\neq P_t(n)\}$。其中 $\{I(n)\}$ 表示实体 n 的交互对象集合，$P_t(n)$ 表示交互的目标实体被分配到的线程，$\{I(n)|P(n)\neq P_t(n)\}$ 表示目标实体与自己不在同一线程的实体 n 的交互集合。因此，式(2)可进一步约简为：

$$\begin{aligned}\min T_{LD} = &\max_{i=1,2,\cdots,m}\left(\sum_{n=1}^{N_m} T_s(i,n)\right)+\\ &\lambda\cdot\sum_{n=1}^{N} num\{I(n)|P(n)\neq P_t(n)\}\\ s.t.\ &P(n)\in[1,2,\cdots,m]\ \ n=1,2,\cdots,N\end{aligned}\tag{3}$$

式中：λ 为交互耗时与交互次数的比例常数，可通过在仿真中插入一定数量的测试交互进行估算。

2.2 METIS 多级 k 路图划分工具

式(3)所示的 PWSF 负载均衡问题是典型的非线性整数规划问题，其可行解数量为 m^N。令线程数为 8，实体数为 1 000，则 $m^N = 8^{1000} \approx 2.037\times10^{90}$。面对如此庞大的求解空间，无论是传统的精确解析方法，还是通用的现代智能优化方法都难以在较短时间内求解。文献[11-12]在 Agent 交互局部性的基础上提出了基于地理空间的负载划分方法，但作战仿真实体可通过 C4ISR 系统与空间任意位置的目标发生交互，因此这些方法在 PWS 中并不适用。

将作战仿真实体看作节点，将实体间的交互看作边，则 PWS 的负载分布可看作为带权重的无向图，PWSF 的负载均衡问题即可看作为图划分问题。

METIS 方法是求解图划分问题的经典方法，在并行仿真[15]、超大规模集成电路(VLSI)设计、交通运输网络系统等领域都有着广泛的应用。但目前，其在国内的介绍和应用还相对较少。对此，本节首先对 METIS 的基本原理做一简要介绍，然后给出其在 PWSF 负载均衡中的应用。

2.2.1 METIS 总体思路

Metis 是美国明尼苏达大学 George Karypis 与 Vipin Kumar 于上世纪 90 年代在前人研究基础上提出的一种高效的多级 k 路图划分方法。它是一种多分辨率的求解方法。如图 3 所示，其求解过程可分为“粗化”(coarsening)、“初始划分”(initial partitioning)和“细化”(uncoarsening)3 步。

给定图 $G=(V,E)$，其中 V 和 E 分别为图的节点集和边集，记 $|V|=N$ 表示节点数量。则 k 路图划分的总体目标为：将 V 划分为 k 个子集 $V(1),V(2),\cdots,V(k)$，使得对任意 $i\neq j$，有 $V(i)\cap V(j)=\varnothing$，且 $\bigcup_i V(i)=V$；同时要求 E 中顶点处于不同分区的边的数量(称之为分区的边割，edge-cut)最小化。用 N 维向量 P 表示 V 的划分方案，其中每一位表示对应节点所属的分区，因此 $P(n)\in[1,2,\cdots,k]\ \ n=1,2,\cdots,N$。

基于以上定义，METIS 粗化阶段的任务可描述为：将原始图 G_0 依次转化为一系列粗化图 $G_1,G_2,\cdots,G_m$，并使得 $|V_0|>|V_1|>|V_2|>\cdots>|V_m|$。初始划分阶段的任务为：计算粗化图 G_m 的 k 路划分方案 P_m，使得各分区的节点数都为 $\frac{|V_m|}{k}$。细化阶段的任务为：将 G_m 的划分方案 P_m 沿着粗化路径 $G_1,G_2,\cdots,G_m$，依次映射回原始图；在细化的过程中，采用启发式算法对映射方案进行微调优化。

METIS 提供了多种粗化、初始划分和细化算法，下面仅对其常用的默认算法做简要介绍，更详细的内容，可参考文献[17]。

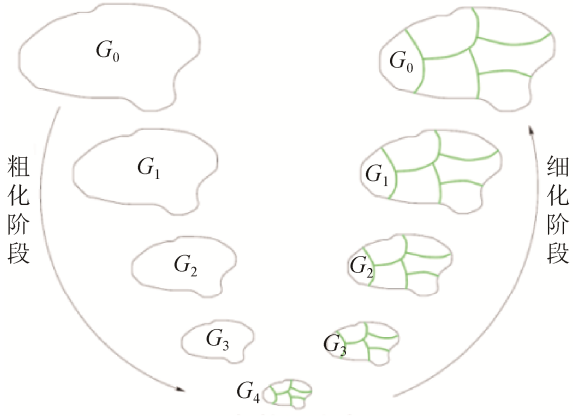


图 3 METIS 多级图划分求解思路
Fig. 3 Main Idea of METIS Algorithm

https://dc-china-simulation.researchcommons.org/journal/vol30/iss6/26
DOI: 10.16182/j.issn1004731x.joss.201806026

第 30 卷第 6 期 系统仿真学报 Vol. 30 No. 6
2018 年 6 月 Journal of System Simulation Jun., 2018

2.2.2 粗化阶段

METIS 通过寻找“匹配”(matching)并聚合匹配节点(matched vertices)的方法实现图的粗化。所谓“匹配”是指一组无共同顶点的边的集合。粗化过程中，未获得匹配的节点被直接拷贝到下一级粗化图。

假设 G_{i+1} 中的节点 v 是由 G_i 中的节点集 V_i^v 聚合得到，则为等效映射原图，应使：

(1) v 的权重为 V_i^v 各节点的权重之和；

(2) v 继承 V_i^v 内各节点与外相连的各边。当有多条继承边相同，即它们都指向同一个目标节点时，则仅保留其中一条，且令其权重为它们的权重之和。

由于粗化的目标是减少图的规模，因此应尽量寻找“极大匹配”进行粗化。所谓“极大匹配”是指图中任意不属于该匹配的边都至少有一个顶点已被匹配。根据寻找匹配的方法不同，极大匹配可能有多种方案。METIS 提供了随机匹配(Random Matching，RM)、重边匹配(Heavy Edge Matching，HEM)、轻边匹配(Light Edge Matching，LEM)和重圈匹配(Heavy Clique Matching，HCM)4 种寻找极大匹配的算法，其中 HEM 为默认算法。

根据经验，边割会随着图的总边权重减小而减小，而理论上粗化图的总边权重等于其原图的总边权重减去其对应匹配的边权重，因此在粗化时采取最大化边权重的匹配策略以减少边割。基于此，HEM 算法的总体思路是：随机遍历 G_i 中的各节点 v；如果 v 尚未被匹配，则选择与其相连、未被匹配且边权重最大的节点 u 进行匹配；如果 u 存在，则将边 (v,u) 加入匹配；如果 u 不存在，则 v 在迭代循环中保持未匹配状态。

2.2.3 初始划分阶段

由于粗化图的规模已经较小，因此在初始划分阶段，很多方法都可在较短时间内给出高质量的划分。METIS 提供了谱二分(Spectral Bisection，SB)算法、Kernighan-Lin(KL)算法、递增图划分(Graph Growing Partitioning)算法和贪婪递增图划分(Greedy Graph Growing Partitioning)算法四种初始划分算法，它们都是通过多次递归二分的方法实现对图的 k 路划分，递归的迭代次数为 $\log_2 k$。其中 KL 为默认算法。

KL 算法的总体思路是：首先对粗化图执行随机二分，使得两个分区具有相同的节点权重；其次在两个分区中寻找两组总权重相同的子节点集，并评估交换它们的分区是否会带来边割的减小，如果是，则执行交换；循环该交换操作，直到再也无法找到能够使边割减少的子节点集为止。这种节点交换的划分方法较大程度地受起始方案质量的影响。对此，实际应用时，通常随机产生多组随机方案，分别对其执行交换算法，并选其中划分效果最好的作为最终的划分方案。

2.2.4 细化阶段

在细化阶段，粗化图 G_m 的划分方案 P_m 会沿着粗化路径 $G_{m-1},G_{m-2},\cdots,G_1$ 被映射回原图。由于 G_{i+1} 中的节点 v 是由 G_i 中的节点集 V_i^v 聚合得到，因此 P_{i+1} 向 P_i 的映射实际上就是为 $u\in V_i^v$ 指派 $P_i(u)=P_{i+1}(v)$。由于 G_i 相比 G_{i+1} 有更多的自由度，因此即使 P_{i+1} 为 G_{i+1} 中的局部最优，P_i 也很可能不是 G_i 中的局部最优。这时，在完成粗化映射后，还需要补充一定的微调优化，以保证每一步细化都有较好的划分效果。常见的微调优化算法都是以 KL 算法为基础。METIS 提供了 KL 细化(KL Refinement)和边界 KL 细化(Boundary KL Refinement)两种微调算法。其中 KL 细化算法为默认算法。

KL 细化算法的总体思路是：以上一级划分方案 P_{i+1} 的映射方案 P_i 作为起始方案，循环执行 KL 算法的交换操作，直到边割不再减小为止。由于 P_i 已经具有了较好的划分效果，因此 KL 细化经过少数的几次循环，很快就能完成收敛。

2.2.5 METIS 应用实例

Karypis 教授已将 METIS 算法封装成开源的工具包，最新的 METIS-5.1.0 版本可在 http://glaros.dtc.umn.edu/gkhome/metis/metis/download 页面下载，其中既提供了独立的可执行程序，也提供了各算法

的 API 函数。对其可执行程序，用户需要按照指定格式录入图数据[18]。

如图 4 所示，一个带权重的图可按以下格式组织数据：第 0 行 3 个数字分别表示图中的节点数，边数和图类型代码(011 表示节点和边都有权重)；其余各行分别记录一个节点的数据，其中，第 i 行的第 0 个数字表示节点 i 的权重，第 $2\cdot j-1$ 个数字表示节点 i 的第 j 条边的目标节点，第 $2\cdot j$ 个数字表示第 j 条边的权重。

METIS 输出的划分结果按以下格式组织数据：每行仅有一个数字，第 i 行的数字表示节点 $i-1$ 被划分到的分区。

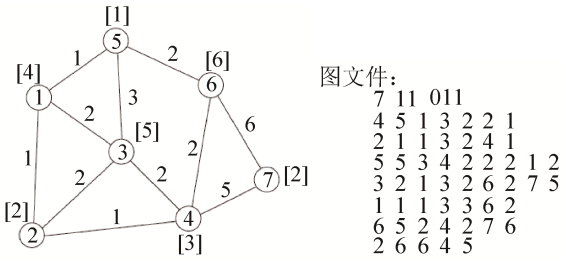
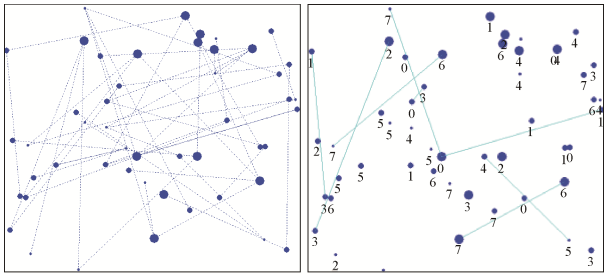


图 4 METIS 图文件格式
Fig. 4 Graph File Format of METIS

图 5 展示了用 METIS 对 30 个仿真实体负载的划分效果，其中实体的计算量服从 1 类分布，各实体都向外发送一个交互，各交互的权重相同。



(a) 原始图 (b) 划分方案

图 5 METIS 图划分效果
Fig. 5 Effect of METIS Graph Partition

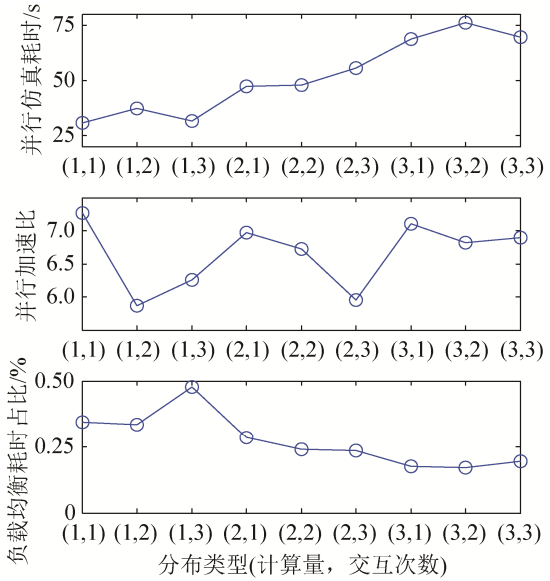
图中不同大小的圆点表示了不同计算量的实体，5(a)图中的虚线表示实体间存在交互，5(b)图中的实线表示图划分后的跨分区交互，圆点下的数字表示实体被划分到的分区。由图可见 METIS 能有效地对节点进行划分，且较大限度地减小了边割。

2.3 实验分析

为全面分析 METIS 方法对 PWSF 的负载均衡性能，我们首先对表 2 所示的 9 种负载分布，在与 1.4 节相同的环境下进行实验；再以(3,3)型的负载分布为例，分别对核心数为 8、实体数 $N=10^2,10^{2.5},10^3,10^{3.5},10^4,10^{4.5}$，及实体数为 10^4，核心数 k=8,16,32,64,128,256,512 时的负载均衡性能进行测试。实验结果如图 6(a)、(b)、(c)所示。

(1) 由图 6(a)可见，采用 METIS 负载均衡方法后，在 9 种负载分布情况下，PWSF 的并行加速比都处于 6~7.5 左右，加速性能理想，且与负载分布类型无明显的模式关系。由图 6(a)可见，METIS 负载均衡耗时相对并行仿真耗时占比很小(0.3%左右)，几乎可以忽略不计。

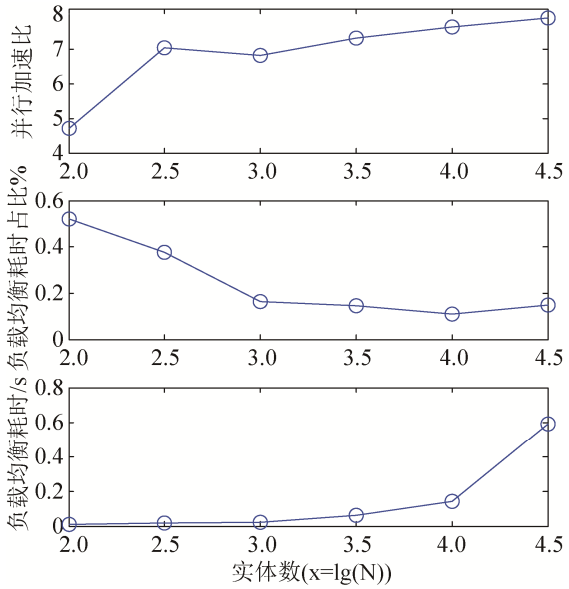
(2) 由图 6(b)可见，当实体数较少时(10^2)，采用负载均衡措施仍然无法克服 CPU 串、并行频繁切换导致的并行加速性能损失；但当实体数超过 10^3 时，并行加速性能即可超过 7，接近 CPU 核心数 8。由图 6(b)和中图可见，随着实体数的增加，METIS 负载均衡耗时虽有所增加，但相对并行仿真耗时的占比却是在持续下降。



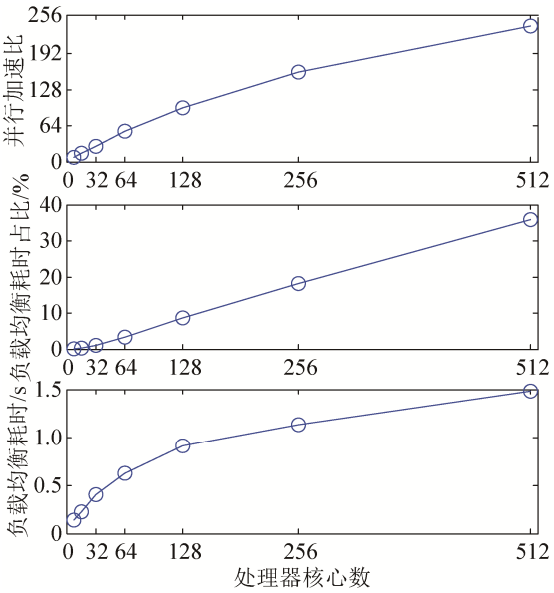
(a) 分布类型的影响

https://dc-china-simulation.researchcommons.org/journal/vol30/iss6/26
DOI: 10.16182/j.issn1004731x.joss.201806026

第 30 卷第 6 期 系统仿真学报 Vol. 30 No. 6
2018 年 6 月 Journal of System Simulation Jun., 2018



(b) 实体数的影响



(c) 处理器核心数的影响

图 6 METIS 负载均衡性能测试

Fig. 6 Load Balance Performance Test of METIS

(3) 由图 6(c)上图可见，PWSF 的并行加速性能随 CPU 核心数近似呈线性递增，这说明了基于 METIS 负载均衡的 PWSF 有良好的可扩展性。需要说明的是，随核心数的增加，跨线程交互的数量不可避免地随之增加，这稍微限制了 PWSF 并行加速比增加的比例。由 6(c)下图可见，负载均衡耗时在增长后仍处于一个很低的水平(约 1.5 s)，而由 6(c)中图可见，随着核心数的增加，METIS 负载均衡耗时相对并行仿真耗时的占比很快达到一个很高水平(40%)，这是因为核心数增加以后，并行仿真总耗时很快由约 130 s 降为 4 s。

根据以上分析，可以得出以下结论：基于 METIS 负载均衡的 PWSF 具有良好的普适性、高效性和可扩展性。

3 结论

随着作战仿真朝着大规模、高精度方向发展，并行仿真必将成为提高仿真速度的主要手段。本文借鉴并行 Agent 仿真理论，建立了并行作战仿真的通用设计框架(PWSF)，采用私有实体链表和动态交互链表技术减少跨线程的通信，并在框架中插入了负载信息采集器为进一步的负载均衡奠定基础。鉴于并行作战仿真与一般并行 Agent 仿真在交互方式上的区别，采用 METIS 工具对仿真负载进行划分，实验结果显示 METIS 工具在 PWSF 负载均衡的应用中具有良好的普适性、高效性和可扩展性。

需要说明的是：本文的研究主要面向一般的大规模作战仿真，在具体的想定场景中，各军兵种的专业仿真模型，如干扰模型、碰撞检测模型、飞行控制模型等可能还具有特殊的并行要求。在实践应用中，根据具体的专业需求对 PWSF 做针对性的优化调整是需要进一步深入研究的内容。

参考文献：

[1] 徐晓东，赵建亭，许春雷. 分导式多弹头弹道导弹并行计算技术研究[J]. 弹箭与制导学报，2013，33(4): 146-148.
Xu Xiaodong, Zhao Jianting, Xu Chunlei. Parallel Technology of Multiple Independently Targeted Reentry Vehicle's Ballistic Trajectory[J]. Journal of Projectiles, Rockets, Missiles and Guidance, 2013, 33(4):146-148.

[2] 陈昊. 基于 Open MP 的并行蚁群算法求解协同空战火力分配[J]. 传感器与微系统, 2013, 32(1): 20-24.
Chen Hao. Parallel ant colony algorithm for solving weapon-target assignment based on OpenMP in cooperative air combat[J]. Transducer and Microsystem Technologies, 2013, 32(1): 20-24.

[3] Macal C M, North M J. Agent-based Modeling and Simulation[C]// Proc. of the 2009 Winter Simulation Conference[S.l.]: IEEE Press, 2009: 86-98.

[4] Paul K Davis, Robert H Anderson. Improving the Compos ability of DoD Models and Simulations[J]. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology (S1548-5129), 2004, 1(1): 5-17.

[5] 杜国红, 韦伟, 李路遥. 作战仿真实体组件化建模研究[J]. 系统仿真学报, 2015, 27(2): 234-239.
Du Guohong, Wei Wei, Li Luyao. Research on Component-based Modeling of Combat Simulation Entity[J]. Journal of System Simulation, 2015, 27(2): 234-239.

[6] J Todd McDonald, Michael L Talbert. Agent-based Architecture for Modeling and Simulation Integration[C]// Proc. of the National Aerospace & Electronics Conference. [S.l.]: IEEE Press, 2000: 375-382.

[7] Lees M, Logan B, Theodoropoulos G. Using Access Patterns to Analyze the Performance of Optimistic Synchronic Algorithms in Simulations of MAS[J]. Simulation (S2310-4791), 2008, 84(10/11): 481-492.

[8] 余文广, 王维平, 侯洪涛, 等. 基于多核 CPU-GPU 异构平台的并行 Agent 仿真[J]. 系统工程与电子技术, 2012, 34(8): 1716-1722.
Yu Wen-guang, Wang Wei-ping, Hou Hong-tao, et al. Parallel Agent-based Simulation on Multi-core CPU and GPU Heterogeneous Platforms[J]. Sysems Engineering and Electronics, 2012, 34(8): 1716-1722.

[9] 余文广, 王维平, 李群. 并行 Agent 仿真研究综述[J]. 系统仿真学报, 2012, 24(2): 245-251.
YU Wen-guang, WANG Wei-ping, LI Qun. Review of Parallel Agent-based Simulation[J]. Journal of System Simulation,2012, 24(2): 245-251.

[10] 姚益平, 张颖星. 集群计算环境下基于复杂网络的社会学仿真负载划分优化算法[J]. 计算机研究与发展, 2011, 48(9): 1759-1767.
Yao Yiping, Zhang Yingxing. An Optimized Partitioning Algorithm for Complex Network Based on Social Simulations on Cluster Computing Platform[J]. Journal of Computer Research and Development, 2011, 48(9): 1759-1767.

[11] 王维平, 余文广, 侯洪涛, 等. 多核 CPU-GPU 异构平台下并行 Agent 仿真负载均衡方法[J]. 系统工程与电子技术, 2012, 34(11): 2366-2373.
Wang Weiping, Yu Wenguang, Hou Hongtao, et al. Load Balancing Mechanism for Parallel Agent-based Simulation on Multi-core CPU and GPU Heterogenous Platform[J]. Sysems Engineering and Electronics, 2012, 34(11): 2366-2373.

[12] Biagio Cosenza, Gennaro Cordasco, Rosario De Chiara, et al. Distributed Load Balancing for Parallel Agent-based Simulation[C]// 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing.[S.l.]: IEEE Press, 2011: 62-69.

[13] Chapman B, Jost G, Van Der Pas R. Using OpenMP: Portable Shared Memory Parallel Programming[M]. London: MIT Press, 2008: 35-49.

[14] 周伟明. OpenMP 编程指南[EB/OL]. http://blog.csdn.net/drzhouweiming/article/details/4093624.
Zhou Wei-ming. A guide to OpenMP Programming [EB/OL]. http://blog.csdn.net/drzhouweiming/article/details/4093624.

[15] 余文广, 王维平, 侯洪涛, 等. 面向并行 Agent 仿真的合成基准测试模型[J]. 系统工程与电子技术, 2012, 34(4): 813-819.
Yu Wen-guang, Wang Wei-ping, Hou Hong-tao, et al. Synthetic Benchmark Model for Parallel Agent-based Simulation[J]. Sysems Engineering and Electronics, 2012, 34(4): 813-819.

[16] AJ Alt, Philip A Wilsey. Profile Driven Partitioning of Parallel Simulation Models[C]// Proceedings of the 2014 Winter Simulation Conference. Florida, USA: IEEE Press, 2014: 2750-2761.

[17] George Karypis, Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs[J]. SIAM Journal on Scientific Computing (S1064-8275), 1998, 20(1): 359-392.

[18] George Karypis. Metis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 5.1.0[R]. Minneapolis, US: University of Minnesota, 2013: 1-32.

https://dc-china-simulation.researchcommons.org/journal/vol30/iss6/26
DOI: 10.16182/j.issn1004731x.joss.201806026