电推进粒子模拟中并行化方法研究综述 *

任军学[1]，潘若剑[2]，毛仁凡[2]，汤海滨[1,3,4]

（1. 北京航空航天大学 宇航学院，北京 102206；
2. 北京航空航天大学 空间与环境学院，北京 102206；
3. 航天器设计优化与动态模拟技术教育部重点实验室，北京 102206；
4. 空间环境监测与信息处理工业和信息化部重点实验室，北京 102206）

摘 要：电推进装置的粒子模拟手段由于其第一性强、物理假设少的特点，可以很好地保留推力器的各种非线性物理过程性质，但计算负担极大，并行计算是一种有效方法计算时间的方法。本文总结了国内外研究现状；分析了粒子算法中并行计算的计算机实现，对并行计算的基础进行了介绍；针对并行计算中的关键技术问题，即粒子部分的并行算法设计、电磁场求解的并行算法设计和计算负载均衡问题进行了概述；最后对电推进装置中并行计算方法进行了总结和展望。

关键词：电推进；粒子仿真；并行计算；电磁场求解；负载均衡

中图分类号：V439+.4 **文献标识码**：A **文章编号**：1001-4055（2023）06-2210070-21

DOI：10.13675/j.cnki. tjjs. 2210070

A Review of Parallelization Methods in Particle Simulation on Electric Propulsion

REN Jun-xue[1]，PAN Ruo-jian[2]，MAO Ren-fan[2]，TANG Hai-bin[1,3,4]

（1. School of Astronautics，Beihang University，Beijing 102206，China；
2. School of Space and Environment，Beihang University，Beijing 102206，China；
3. Key Laboratory of Spacecraft Design Optimization and Dynamic Simulation Technologies，
Ministry of Education，Beijing 102206，China；
4. Laboratory of Space Environment Monitoring and Information Processing，
Ministry of Industry and Information Technology，Beijing 102206，China）

Abstract：Because of the first principles and fewer assumptions，the particle simulation can well preserve various nonlinear physical process properties of electric propulsion devices. However，the computational burden is great，and parallel computing is an effective method to reduce the computation time. The state-to-art status of domestic and international research on the parallel computing is summarized in this paper. The computer implementation of parallel computing in particle simulations is analyzed and the basic knowledge of parallel computing is introduced. The key technical issues in parallel computing，i.e.，parallel algorithms of particle advances，solving methods of electromagnetic field，and computational load balancing，are figured out. Finally，a summary is made to outlook the parallel computing methods in electric propulsion devices.

Key words：Electric propulsion；Particle simulation；Parallel computing；Electromagnetic field solver；Load balancing

* **收稿日期**：2022-10-21；**修订日期**：2022-12-12。

作者简介：任军学，博士，副教授，博士生导师，研究领域为电推进技术。

通讯作者：汤海滨，博士，教授，博士生导师，研究领域为电推进技术。E-mail：thb@buaa.edu.cn

引用格式：任军学，潘若剑，毛仁凡，等. 电推进粒子模拟中并行化方法研究综述[J]. 推进技术，2023，44(6)：2210070.（REN Jun-xue，PAN Ruo-jian，MAO Ren-fan，et al. A Review of Parallelization Methods in Particle Simulation on Electric Propulsion[J]. *Journal of Propulsion Technology*，2023，44(6)：2210070.）

1 引 言

空间电推进相比于传统化学推进不同，通过电能到粒子动能的最终变换，有高比冲、高效率的特点[1-2]，自被发明以来被广泛应用在轨道保持、轨道提升等任务中，将来还可被应用到深空探测、载人飞船转移等任务上[3-5]，具有广阔的前景。

在电推进装置的研究中，实验手段作为主要的研究手段，被广泛的应用在等离子体参数测量、宏观性能评估等方面。但由于技术的限制，实验手段也存在一些缺陷：如难以直接获得的物理过程、成本过高等问题。此时仿真手段可作为实验研究的补充，实现揭示物理机制、指导实验方案、优化推力器设计的目的。例如空心阴极作为重要的电推进装置部件，其孔径难以放入接触式测量设备，而非透明的外壳也使得光学诊断设备无法工作，故若要研究阴极内部的能量分布、发射体烧蚀过程等物理机制，仿真手段是一种不可或缺的研究方式；再如研究霍尔推力器磁场优化设计、推力器腐蚀与寿命等问题，仿真手段可以避免高额的实验费用、缩短研究时间。

仿真手段按照对不同组分的处理方式，一般可以分为三种手段：将所有组分都看作是粒子的粒子仿真方法，将组分都看作是流体的磁流体方法，以及综合上述两种方法的混合方法。相比之下，粒子算法由于其第一性强，物理假设少，可以获得更丰富的非线性过程，基于统计的计算方法可以得到更多推力器中等离子体的细节信息。

但粒子方法最主要的缺点也非常明显，即计算效率非常低，这是由于对于N个粒子而言，其收敛性质和$N^{-1/2}$成正比[6]。粒子算法的计算负担大部分来自两部分：一是空间上存储与追踪大规模的粒子的运动所需要的内存和算力非常高，一般所需要追踪的宏粒子（一个粒子代表若干真实粒子）数目可达10^5~10^7甚至更多。二是电推进中等离子体的特征尺寸小、特征时间短[7]，等离子体的特征时间主要由德拜长度λ_{De}决定，这导致仿真较大的空间尺寸需要非常稠密的网格划分；等离子体的特征时间主要由电子的振荡频率ω_{pe}决定，考虑到算例的收敛性，以及需要体现宏观物理现象（如湍流、不稳定性等），仿真的时间步一般在10^7步甚至更多。以典型SPT-100霍尔推力器为例，其最高等离子体密度大约在$10^{19}\mathrm{m}^{-3}$，电子温度在20~30eV，其德拜长度λ_{De}约为$10^{-5}\mathrm{m}$，电子的振荡频率ω_{pe}约为$2\times10^{11}\mathrm{Hz}$。故若要仿真放电通道和近场羽流区域80mm×70mm的区域，算例需要追踪的真实粒子总数目（电子，离子和原子）约为4×10^{15}个；若计算网格是均匀正交的，则网格需取8000×7000，时间步长约取$5\times10^{-12}\mathrm{s}$。推力器点火时间约为$10^{-4}\mathrm{s}$，而低频振荡一般在10~100kHz（即周期为$10^{-5}$~$10^{-4}\mathrm{s}$），所以为了仿真推力器点火成功并体现低频振荡特性，需要计算超过10^8时间步。在实际仿真中，研究者采用了一系列的方法来直接降低计算量。一方面，德拜长度λ_{De}与当地等离子体数目相关，所以等离子体中心区域和鞘层区域存在的高密度区导致了当地的德拜长度小，若采用均匀网格则会使全局网格划分过密，此时可采用非均匀网格或动态网格划分方法进行局部网格加密。另一方面，可以使用多种数值加速方法来减少计算时间，如人工降低重粒子质量[7]、人工增大真空介电常数[8-9]、自相似缩比方法[10]等，但这些方法本身破坏了推力器中鞘层等物理效应，在使用时需要较为谨慎，以免物理问题失真。

然而，粒子方法粒子之间、区块之间的相对独立性又给研究者从并行算法的角度来进行计算加速提供了可能。并行算法加速的概念是相对于并行算法提出的，即一般的仿真计算是使用计算机的单处理器将程序从头至尾顺序执行的，而并行计算加速则利用多处理器、多计算核心等手段，将任务分解成多块之后同步进行，其自然会有更好的计算性能[11]。目前，随着摩尔定律逐渐失效，计算机单处理核心的性能提升已经逐渐达到瓶颈，并行计算就显得更加重要。

并行计算的难点在于，相对于串行算法，程序的编写对研究者的编程能力要求更高，涉及到算法设计和框架构建的诸多难点，例如多种并行手段、编程语言的选择问题，粒子推动和电磁场求解的并行算法设计问题，拓展性和易用性问题等。

本文介绍国内外相关研究现状，从电推进装置粒子方法的并行化框架设计思路，阐述从串行代码到并行代码所需要进行的算法设计。对之前研究进行总结，并且对未来并行化方法和框架设计进行展望。

2 国内外研究现状

2.1 国外研究现状

国外粒子方法的并行化研究远远领先于国内。许多国外机构都对代码进行了并行化处理，值得注意的是，由于一些混合模型中，粒子部分也使用了并

行手段，所以本文也将同时介绍这部分的工作。

美国麻省理工学院的Szabo[7]开发了MIT-PIC代码用于霍尔推力器的仿真，其课题组的Fox[12]在串行代码上进行了并行化处理；同在麻省理工学院的Roy等[13]使用了并行化的代码仿真了离子推力器的三维羽流反流。密歇根大学的Cai等[14]在研究霍尔推力器的三维羽流特性中使用了并行手段，Fergason[15]通过并行化代码模拟了离子推力器羽流；日本航天局(JAXA)的Cho等[16]开发的霍尔推力器粒子仿真模型，以及韩国世宗大学的Lee[17]开发的研究羽流对航天器影响的计算模型，都使用了并行化加速方法。

另外，国外的研究机构研究出了诸多成熟的并行框架，并产生了大量应用。加州大学洛杉矶分校的Decyk课题组早在20世纪90年代就开始了针对通用等离子体仿真的并行框架研究，后逐渐发展成了GCPIC(General concurrent PIC)[18]框架。该框架主要被用于空间等离子体、激光等离子体等领域的研究。在后续的发展中[19-20]进一步形成了新版的UPIC[21]，而在UPIC基础上衍生开发出的等离子体模拟软件众多，包括由Huang等[22]开发的应用于粒子加速器中等离子体研究的QUICKPIC，以及与电推进相关的DRACO。弗吉尼亚理工大学的Wang等[23]基于GCPIC和UPIC首先开发了一种三维并行等离子模拟代码，之后逐渐发展为电推进并行算法求解库DRACO[24-26]，其课题组在此基础上进一步完善了代码[27-28]，将浸入式有限元(Immersed Finite Element，IFE)用于电推进羽流的求解中[24]。DRACO被广泛应用于多种电推装置的模拟，例如：离子推力器放电室仿真[28]、霍尔推力器的羽流仿真[29]、磁喷管的羽流仿真[30-31]等。

加州大学伯克利分校研制的OOPIC[32]和拓展的Xoopic[33]被应用于霍尔推力器[34-35]，离子推力器[36-38]，双层螺旋波推力器[39]等电推装置的仿真中。怀特大学的Mahalingam[9]在XOOPIC基础上重写了一些求解器，将代码运用在了NASA的离子推力器上[40-42]；科罗拉多大学研发的VORPAL框架[43]后面演变为了商业软件VSim，被应用在了阴极弧推力器[44]、会切霍尔推力器[45]等电推装置的仿真中。

伊利诺伊大学香槟分校研究的CHAOS框架被应用在离子推力器羽流-航天器帆板相互作用的分析中[46-47]；伍斯特理工大学开发了EUPIC软件[48]，该软件是并行化的，适用对象为完全电离的等离子体，被用来模拟电推进羽流对立方星的影响；斯坦福大学开发了带有用户界面的霍尔推力器羽流软件BEPPA[49]，该模型中电子采用流体模型，重粒子(原子和离子)采用并行化的粒子算法；德国不莱梅大学的picFOAM[50]基于开源的OpenFoam框架修改，被用于霍尔推力器羽流仿真中。

2.2 国内研究现状

国内方面，使用并行化的粒子方法进行电推力器仿真研究的主要有以下单位。

北京航空航天大学的汤海滨课题组发展了多种并行手段，Ren等[51]和仇钎[52]使用了GPU加速手段对离子推力器的放电室进行了加速计算；李卓恒[53]首次将实验室串行代码JLPP2.5进行并行化处理，并被应用于圆柱形霍尔推力器，Pan等[54]在其基础上修改了原子分布求解器。北京航空航天大学的蔡国飙课题组[55-58]对多种电推相关的研究使用了并行化加速手段，其中多项研究被用于羽流仿真上；Liu等[59]在对小离子推力器放电腔调的仿真中也进行了并行优化。

中科院力学所[24-26]主要继承了其在南加州大学时Wang的计算方法，将DRACO应用在了磁碰管模拟中[60]，其追踪的粒子规模甚至达到了惊人的2亿个之多[61]；哈尔滨工业大学(深圳)的曹勇课题组同样继承了Wang的处理方式，将IFE-PIC应用到了霍尔推力器羽流模拟中[62]，并且进一步发展成了隐式求解的IFE-PIC模型[63]。

西安交通大学的孙安邦课题组[64-65]主要在研究离子推力器的栅极束流引出机制中使用了并行化加速方法；西北工业大学的Fu等[66]在研究ECR离子推力器时提及了并行计算方法，将一个算例的计算时间控制在了30h；国防科技大学吴建军课题组[67-68]在对电推进羽流的直接蒙特卡洛碰撞(DSMC)建模上采用了并行化加速方法。

国内并行算法框架的研究主要有两个单位。电子科技大学金晓林课题组[69-70]开发的BUMBULEBEE框架被应用在激光等离子体等领域上，其二维求解器被用来模拟离子推力器[71-72]。另外，北京航空航天大学的汤海滨课题组开发了一种面向多种电推力器的通用架构GPIC(general PIC)，已经被成功应用在阴极仿真[73]、霍尔推力器仿真[74]的并行化上。

国内外主要的粒子并行研究如表1所示。

为了表格简洁，表1中采用了若干缩写，表2是对表1中缩写的解释。

Table 1　Summary of parallelization research

Researcher/Framework	Institution	Application on electric propulsion
Fox	MIT	SPT[12]
Roy	MIT	Plume of IT[13]
Cai	UM	3D plume of HT[14]
Fergas	AeroCorp	Plume of IT[15]
Cho	JAXA	SPT[16]
Lee	Sejong	Plume-craft interaction[17]
GCPIC[18]/UPIC[21]	UCLA	DRACO[23]
DRACO[23-24]	VT/USC/JPL	Plume of IT[28], plume of HT[29], MN[30-31]
OOPIC[32]/Xoopic[33]	Berkeley	HT[34-35], IT[36-38], DLHT[39]
Mahalingam	Wright	IT[9,40-42]
VORPAL[43]/VSim	CU	VAT[44], HEMPT[45]
CHAOS	UIUC	Plume of IT-craft interaction[46-47]
EUPIC	WPI	Plume-CubeSat interaction[48]
BEPPA	Stanford	Plume of HT[49]
picFOAM	Bremen	Plume of HT[50]
Tang et al.	BUAA	IT[51-52], CHT[53-54]
Cai et al.	BUAA	Plume of IT[56,58], plume of HT[57], IT[59]
Hu	IM, CAS	MN[60-61]
Cao et al.	HITSZ	Plume of HT[62], HT[63]
Sun et al.	XJTU	Grids of IT[64-65]
Fu	NPU	ECR IT
Wu et al.	NUDT	DSMC simulation[67-68]
BUMBLEBEE[69-70]	UESTC	IT[71-72]
GPIC[73]	BUAA	HC[73], HT[74]

Table 2　Abbreviations in table 1

Abbr.	Full name	Abbr.	Full name
MIT	Massachusetts Institute of Technology	BUAA	Beihang University
UM	University of Michigan	IM,CAS	Institute of Mechanics, Chinese Academy of Science
AeroCorp	Aerospace Corporation	HITSZ	Harbin Institute of Technology, Shenzhen Graduate School
JAXA	Japan Aerospace Exploration Agency	XJTU	Xi'an Jiaotong University
Sejong	Sejong University	NPU	Northwestern Polytechnical University
UCLA	University of California, Los Angeles	NUDT	National University of Defense Technology
VT	Virginia Polytechnic Institute and State University	UESTC	University of Electronic Science and Technology of China
USC	University of southern California	SPT	Stationary Plasma Thruster
JPL	Jet Propulsion Laboratory	IT	Ion Thruster
Berkeley	University of California, Berkeley	MN	Magnetic Nozzle
Wright	Wright State University	DLHT	Double Layer Helicon Thruster
CU	University of Colorado	VAT	Vacuum Arc Thruster
UIUC	University of Illinois at Urbana-Champaign	HEMPT	High Efficiency Multistage Plasma Thruster
WPI	Worcester Polytechnic Institute	CHT	Cylinder Hall Thruster
Stanford	Stanford University	DSMC	Direct Simulation of Monte Carlo
Bremen	University of Bremen		

3　并行框架设计

3.1　粒子算法的并行性分析

电推进装置的本质上是对低温等离子体进行仿真，在粒子方法中，常用粒子网格法（Particle in Cell，PIC）来追踪和计算粒子信息和电磁场信息；等离子体内粒子的相互碰撞可使用不同类型的碰撞来进行模拟，主要的方法包括蒙特卡洛碰撞法（Monte Carlo

Collision，MCC）和直接蒙特卡洛碰撞法（Direct Simulation of Monte Carlo，DSMC）。MCC和DSMC方法都基于随机变量和随机抽样，其理论基础都是概率论和统计方法中的中心极限定理，对于有碰撞的PIC-MCC或PIC-DSMC模型，基本的仿真流程可以由图1表示。

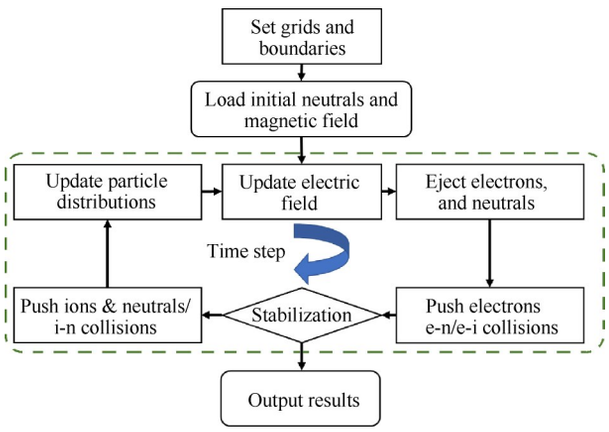

Fig. 1　Scheme of the numerical algorithm of particle simulation for electric propulsion devices[75]

在整体仿真开始前，需要进行一些静态求解和初始条件确定，如时间和空间步长设置、初始电磁场分布、初始粒子分布、边界条件的设定等，这些计算和设定在文献[7,9]中有详细的叙述。在之后的每一个时间步长内，仿真通过上一时间步长结束时网格点上的电磁场信息求解粒子受力情况，进而求解运动方程，在计算域中推动粒子；粒子达到新位置后会产生新的电荷密度和电流，通过求解Maxwell方程组可以得到新的电磁场分布，从而在步长结束时计算出下一步长粒子受到的电磁力；如此周而复始，粒子的运动过程就得到了仿真。

同样的，对于无碰撞等离子体，其粒子追踪的仿真部分与有碰撞的等离子体完全一致，只是在每个时间步长内不需要考虑碰撞部分。

在程序中，可以进行并行的主要有两大部分：一是粒子部分，二是电磁场求解部分。粒子部分的处理需要利用动力学方程追踪和计算巨量的粒子运动和碰撞，但粒子在电磁场中的运动是相互独立的，且粒子之间的碰撞也可以相互解耦，所以具有一定的天然可并行性；随着计算区域的扩大，电磁场计算求解的计算量也将成为一个巨大的负担，但电磁场求解最终可以在算法上等价于对线性方程组的求解，这部分也可以实现并行化。所以，之后研究人员在算法上的加速研究主要也针对这两大方面的相关算法进行优化，这些优化将在第4节中详细介绍。

3.2　程序设计中思想：面向对象或面向过程

在一般的软件中，代码的开发者和使用者的身份是分开的。但是电推进的粒子仿真属于科学计算范畴，代码的开发者往往也是用户。随着仿真代码日渐复杂，研究者需要进一步优化软件架构的设计，力求做到下列几点：

（1）用户友好，即对代码并不熟练的科研人员也可以快速上手，通过修改若干定义的数值进行仿真和调试，最好有图形化界面。

（2）拓展性，即开发者如要增加一个新功能，不需对原来的代码结构进行大量修改。

（3）复用性，即开发者抽象出重复使用的功能，避免重复书写。

（4）封装性，即基本代码调试完毕之后，封装成函数等载体，避免被错误修改。

（5）高性能，即需要对算法和数据结构进行优化，以期达到更高的计算速度。

在20世纪80年代之前，早期的粒子算法（主要是PIC-MCC算法）还并未使用在电推进领域，并且还处于串行计算阶段。此时的代码都只模拟物理机制简单、空间维度较低的等离子体过程[76-77]，所以涉及到的物理模型少、计算量小。这些程序通常采用传统语言，如FORTRAN的早期版本和C语言进行书写，采用的是面向过程的编程思想，即程序从头到尾按照物理过程的时间顺序进行书写。面向过程的代码较为简单，通常比较好理解，但是程序的针对性太强；并且由于缺少用户界面，需要专家才能运行代码。

在后续研究中，随着空间维度增多、物理过程变复杂，这些面向过程的程序逐渐转化为了包含10^4至10^6行的大型代码。为了解决针对性太强的问题，研究者们逐渐开始开发一些通用化的代码，包括美国空军科学研究办公室（Air Force Office of Scientific Research，AFOSR）研制的MAGIC代码[78]，康奈尔大学设计的ISIS通用计算平台[79-80]等。这些代码提供了有限的参数化的灵活性，是对历史代码的修补。然而，随着建模所需的模型和算法数量的进一步增加，面向过程的编程结构还是因为拓展性差、耦合强、封装弱等问题，难以进行模块化升级。

之后的研究者逐渐意识到了上述的问题，开始逐渐尝试使用面向对象的思想来实现代码，加州大学伯克利分校设计的一维等离子体通用代码PDx1[81]开始采用C语言中的结构体作为面向对象的过渡，Forslund等[82-83]开始使用C++语言编写UNXI系统上

的 PIC 算法，Gisler 则使用 Object Pascal 语言书写粒子追踪程序。

加州大学伯克利分校的 Verboncoeur 等[32]在 1990 年开始着手将通用等离子的粒子算法进行面向对象的转化，C++中的模板、类、对象等概念都被应用到了实际仿真代码中，程序将数据结构主要分为了标量、二维向量、三维向量和链表，之后建立了 Grid，Fields，Boundary，Particle 等类，实现了代码中主要物理参数的对象化，最终发展成了 OOPIC 这一框架，后续演变成了 XOOPIC 框架[33]。图 2 显示 OOPIC 中类的继承关系。该框架后续被用到了诸多研究当中，如霍尔推力器[34-35]，离子推力器[37-38]等。

现在针对电推进程序书写的并行代码，主要有三种设计思想，即面向过程思想，面向对象思想和多层设计思想。

面向过程的代码，其一般是由串行代码直接修改而来的，其中会有一些结构体等面向对象的设计思想，但是整体是依靠 C 语言、Fortran 语言的早期版本设计。如北京航空航天大学的 JLPP2.5 代码的串行版本被广泛应用于离子推力器[75,84]、磁喷管[85-87]、圆柱形霍尔推力器[88]、阴极[89-91]的仿真中，后通过 MPI 等并行手段，逐渐成为并行版本[53-54]。类似的还有：北京航空航天大学的蔡国飙等[55]针对电推力器的羽流模拟研究，西安交通大学孙安邦等[64-65]针对等离子体在粒子推力器中栅极引出机制的研究，国防科技大学张海红和张志远[67-68]针对 DSMC 算法的优化研究，这些工作都是在代码的整体或者局部进行并行优化，并没有对代码本身进行重新设计。

面向对象的代码需要对代码进行重构，使之具备易拓展、易复用、高封装等特性，现阶段明确提及自身是并行化的面向对象设计的研究有：斯坦福大学的 BEPPA 和 The Aerospace Corporation 公司的 Fergason[15]开发的离子推力器羽流仿真软件等。

第三种设计思想为多层设计思想。其目的在于根据不同层级的代码需实现的目的进行分层设计。偏向于计算机底层的代码需要更高的可执行性能，用户操作极少；反之，偏向于用户层级的代码，由于用户操作极多，故需偏向于用户友好。加州大学洛杉矶分校研制的 UPIC[21,92]采用了如下的架构：最底层使用 Fortran77 编写，主要有两个原因，一是历史代码的继承问题，二是测试显示 Fortran77 的并行性能最高；中间层使用了 Fortran90 对最底层进行包装，使之成为一个库，修复了一些诸如指针效率低等 Fortran77 的劣势；由于 Fortran90 并不完全支持面向对象，用户层则使用了 C++或是 Fortran2003 进行进一步的封装，用户此时无需关心底层代码的性能问题，而编写新程序也非常简单。这种思想可以兼顾底层性能和用户友好性，但对代码设计者有很高的要求。

3.3　并行手段的选择

3.3.1　并行计算的硬件平台

并行计算的硬件平台有着多种分类方式：(a) Flynn 分类法[93]根据指令流和数据流可将计算机分类

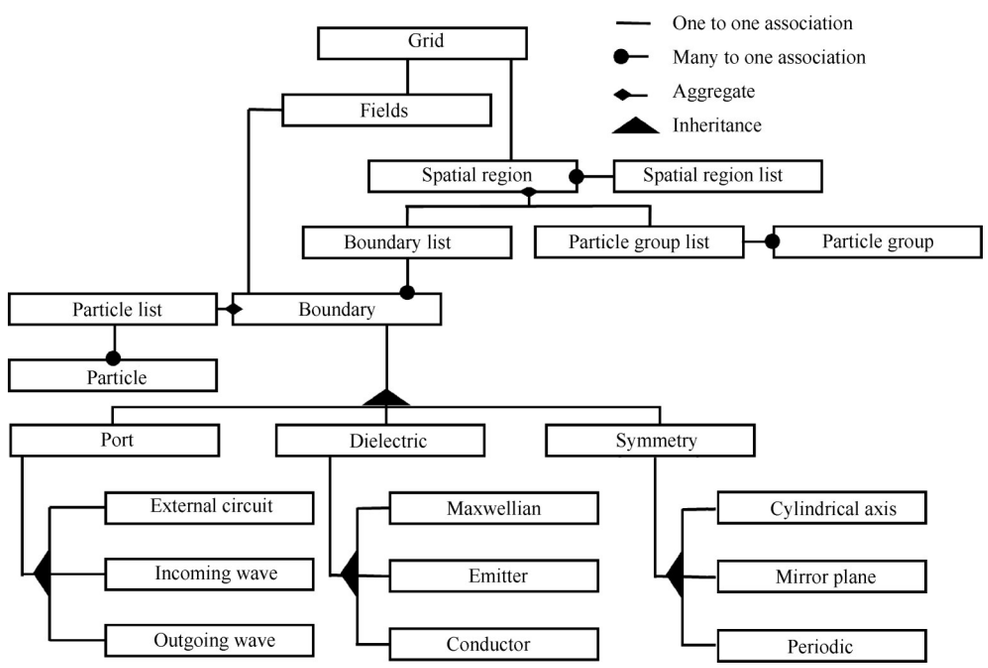

Fig. 2　Class hierarchy for the OOPIC code[32]

为四种：单指令流单数据流（SISD）、单指令流多数据流（SIMD）、多指令流单数据流（MISD）、多指令流多数据流（MIMD），常见的并行系统主要为SIMD与MIMD两大类。(b)按照存储体系，可将并行机分为分布式存储与共享式存储两大类。(c)从体系结构看，并行机目前主要有集群（Cluster）、星群（Constellation）和大规模并行机（Massively Parallel Processing）三大类。按照上述分类的依据，硬件平台主要分为以下几种：

(1)向量计算机：SIMD，共享内存的并行系统。传统中央处理器（Central Processing Unit，CPU）对单独的数据元素或者标量进行操作，被称为标量计算机；而向量计算机同时拥有标量运算指令和向量运算指令，其实质是在CPU单个核心上用一条指令同时处理多个数据。

(2)多核处理器和基于CPU的集群系统：多核处理器是一种MIMD、共享内存类型的并行系统。一个多核处理器在一块芯片上有多个CPU或者计算核心，它们彼此共用同一个内存。而基于CPU的集群系统（如图3所示）是一种MIMD，分布式内存的并行系统。每一个计算节点都为一个多核处理器，每个节点都拥有自己的私有内存，各个计算节点内部的核之间共享内存，而在节点之间利用网络进行显式通信。

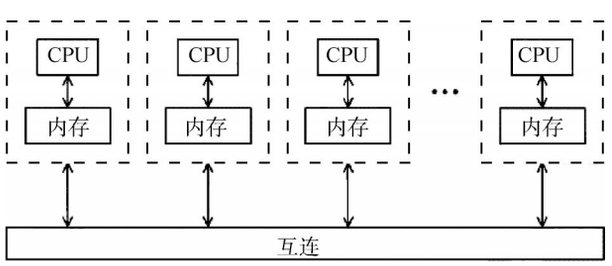

Fig. 3　Structure of a CPU-based cluster

(3)GPU技术：现代图形处理单元（Graphics Processing Unit，GPU）的同期计算能力往往数倍于CPU[11]。随着技术的发展，GPU不再仅限于图像计算方面，逐渐进入通用计算领域，随之发展起来另一种并行计算体系架构——单指令流多线程（SIMT）。CPU近年来发展缓慢，GPU却仍有很大的提升空间。CPU和GPU的各自的特点在于：CPU计算主频高、逻辑计算能力强，更加擅长操作系统以及各种通用化程序，对于循环、分支、逻辑判断等复杂的计算具有更多优势；而GPU由于IPC（每个时钟周期的指令数）高、计算核心多、并发能力强等优势，有着浮点计算性能高的优势。简而言之，CPU比较适合涉及复杂逻辑和通用用途的计算，如粒子模拟中碰撞处理涉及到的碰撞对处理、碰撞种类判断等；而GPU则适合简单数据的重复计算，如粒子模拟中大量的、相互独立的粒子的推动模拟，以及需要求解大规模矩阵方程的电磁场求解中。CPU与GPU的区别见图4。

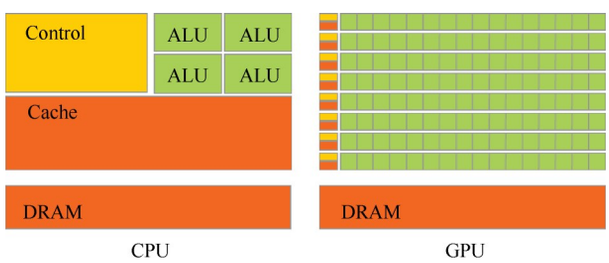

Fig. 4　Architecture of CPU and GPU[70]

(4)异构系统：现阶段，CPU加GPU的异构系统已成为超级计算机发展的一个方向。全球TOP500的超级计算机大多采用异构系统，国外有美国的“Titan”，“Summit”等，日本的“京”、“富岳”等；国内上榜的有“天河一号”、“天河二号”、“神威·太湖之光”、“曙光星云”等。云计算近年来发展迅猛，其本质就是异构化的大规模分布式计算资源。国际主流云计算平台有亚马逊云服务、微软云计算等；国内主流云计算平台有如阿里云、腾讯云等。

3.3.2　并行计算的软件支持

并行采用的软件支持与硬件平台密不可分，和前述的硬件平台分类有着较强的对应关系。

(1)SIMD的应用：向量计算

大多数现代CPU都是向量处理器，所以保留了向量运算的功能，可通过特殊指令集调用，实现对特定数据结构的数据并行。然而，向量计算并行化编程复杂，适用范围窄，所以只能作为辅助手段而不是主要的并行方法。

(2)MIMD共享内存方式：Pthread和OpenMP

对于多核CPU，既可使用操作系统自带的多线程库编程（如Linux自带的Pthread），也可使用针对对称多处理器结构（Symmetric Multi-Processor，SMP）节点集群的跨平台线程库（如OpenMP）。POSIX线程（Pthread）的API属于低层原语且基于命令，标准化程度低，所以使用也较少。OpenMP是一种支持共享内存编程模型的API，可使用FORTRAN，C语言以及C++语言在共享地址空间计算机上编程。使用OpenMP并行优化时将线程处理的大部分工作留给

编译器，大大降低了代码的复杂性。

伍斯特理工学院的EUPIC软件[48]使用了OpenMP进行了局部的算法加速。北京航空航天大学的GPIC框架[73]的并行化代码中使用到了OpenMP进行多线程优化，来处理粒子推动和电磁场求解模块。

(3)MIMD分布式内存方式：MPI

分布式内存系统由网络连接的核-内存对的集合组成，与核相关联的内存只能由与之关联的核才能访问，若想要得到全局结果，需要在不同内存之前进行消息传递。研究者将使用消息传递的实现称为消息传递接口(Message-Passing Interface，MPI)[94]。MPI定义了一个可以被C，C++和Fortran程序调用的函数库，基于此标准研制的各类并行库(如MPI-CH[95]，OpenMPI[96])可以在不同处理器、不同操作系统下编译运行，在调用命令上是相同的。因此，同一份MPI代码可以不经修改地在多核CPU、集群或是超级计算机上运行，增强了代码的通用性和可移植性。

实际上，多数追求高性能的粒子算法的并行手段都采用了或者包含了MPI算法。加州大学伯克利分校研制的OOPIC/XOOPIC[97]使用了基于C++语言的MPI手段，对粒子和电磁场求解进行了并行，并且运行在了Linux系统的计算集群上；日本JAXA的Cho等[16]使用MPI算法对霍尔推力器的PIC-MCC模拟进行了并行化，程序运行在了12核心CPU，主频为3.4 GHz的工作站上；Fox[12]在对霍尔推力器粒子模拟并行化时，使用了MPICH库，并采用32核的康柏计算机作为工作站进行计算；怀特大学的Mahalingam[9]在仿真离子推力器时，也使用MPI方法，在一个10核心处理器上进行了算法加速；北京航空航天大学的王虹月[56]在仿真离子推力器羽流时，通过MPI方法将算例在32核计算机上并行加速；西安交通大学的孙安邦等[64]，韩国世宗大学的Lee[17]也在各自代码中使用了MPI方法加速。北京航空航天大学的圆柱形霍尔[54]的并行化代码中使用到了MPI方法，在阿里云32核商用并行机上实现了并行加速。

值得注意的是，目前在超级计算机上运行的大规模并行程序均是基于MPI来研制的。如北京应用物理与计算数学研究所研制的面向结构网格并行编程框架[98-100](J Adaptive Structured Meshes Applications Infrastructure，JASMIN)，其在“天河2号”和“神威·太湖之光”等超算上均实现了百万核级别的并行计算。这为电推进方面的粒子并行的规模提供了参考。

(4)基于GPU的加速方法

对于GPU，可以使用OpenCL或CUDA等面向异构系统的编程库来实现计算的加速。OpenCL[101]是一个基于C语言标准化的开放标准，可提供跨平台的并行计算API，但是在电推进领域的使用较少。

CUDA是英伟达公司提出的一款将GPU用于数据并行计算的软硬件体系，全称为Compute Unified Device Architecture(统一计算设备架构)[102]，是一种C语言的拓展。传统的GPU计算需要将并行计算的问题转化为图形渲染的求解，再通过接口使用GPU的计算资源，而CUDA可以让程序员使用C/C++/Fortran/Python等高级语言进行编程，通过一些基本关键字的形式来实现并行[103]。CUDA使用了统一的处理架构体系，使得计算资源可以被充分的利用，并且引入了对于存储器的片内内存共享机制，这些都是之前GPU并行手段不存在的。CUDA编程模型采用了双端结构，使用CPU作为主机端，GPU作为协处理器，主机端和设备端协同工作，共同完成计算任务：逻辑性较强的问题和串行问题在CPU端进行，而并行处理任务因为高度的线程化则在GPU端进行计算[104]。CUDA软件层的示意图如图5所示。

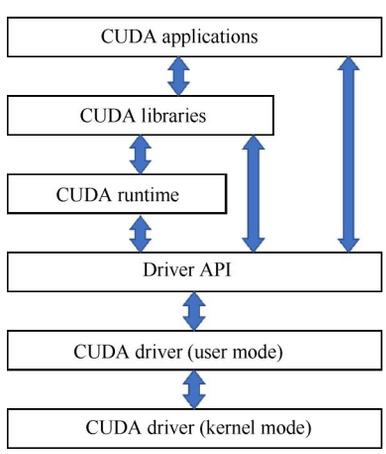

Fig. 5　Softrware structure of CUDA[105]

由于近来GPU算例的稳步提升和CPU算例的止步不前，越来越多的粒子并行程序选择使用CUDA作为主要或者部分加速手段。北京航空航天大学的Ren等[51]和仇钎[52]就使用了CUDA方法，使用GeForce 9400GT对离子推力器的计算进行加速；而其他研究者也多将CUDA加速作为混合加速手段中的一种使用在算例中。

(5)混合加速的方法及其他

在并行计算的发展过程中，为了追求更高性能，在集群的多台计算机之间采用多种加速手段，在单

台计算机内的多核处理器上实现共享内存并行方式(OpenMP手段),在集群内部的计算机之间引入分布式内存并行手段MPI,另外又可以引入GPU加速的方法。这种混合的加速方式结合了多种加速方法的优点,进一步提升了并行的性能。在粒子仿真计算中,有多种应用:德国亥姆霍兹学会和德累斯顿科技大学联合开发的PIConGPU[106]采用三维运动模型仿真了多种等离子体,代码使用了GPU和CPU混合加速的方法;加州大学洛杉矶分校的通用的并行PIC框架GCPIC[18]和UPIC[21,92]可在多核CPU,GPU,集群上运行;Decyk等[107]研究了分别基于CUDA和基于OpenMP的两种并行PIC代码;CHAOS框架[46]可以实现对CUDA和MPI两种加速方法的调用;EUPIC[48]使用了CUDA和OpenMP的混合并行方法;电子科技大学的BUMBLEBEE框架[108]主要使用CUDA和MPI进行加速。

另外,还有一些使用较少的并行加速方法,如电子科技大学的BUMBLEBEE[109]使用了TBB(Threading Building Blocks)方法优化了部分求解器。TBB是一种由英特尔公司提出的多核心并行加速方法,可以和MPI和OpenMP方法兼容,在直流磁控溅射放电的研究中也有所使用[110],但在电推进粒子并行算法中使用较少。

针对上述并行方式,本文对常用的几种并行编程方案从硬件层面、并行粒度、性能、可移植性和可拓展性等方面进行了评估,结果见表3。

纵观多种并行手段,OpenMP作为线程级别、自动并行优化的加速手段,适合小规模集群或是多核计算机,方便修改;MPI方法由于其强大的拓展性和可移植性,加速性能上限非常高,适合拥有大规模计算资源的研究者;CUDA方法计算性能高、发展潜力大,已被越来越多研究人员使用,但其高学习成本成为了制约科研人员代码开发的障碍。若想追求极致性能则可运用多种加速方法混合,但代码的书写和优化难度高,适合拥有丰厚仿真底蕴的课题组。

4 算法的并行化实现

4.1 粒子算法的并行化

电推进粒子算法中涉及到粒子追踪的算法主要分为几块:

(1)粒子分发:即新进入计算域或新产生的粒子的速度、位置分配问题。

(2)粒子推动:粒子在电磁场作用下的运动问题。

(3)粒子碰撞:粒子之间或粒子-壁面之间的相互作用问题。

对于上述的算法,研究人员主要开发出了两种粒子算法的并行化思想:区域分解和粒子分配。

区域分解算法:1988年哈佛大学的Mankofsky等[111]开创了区域分解式PIC程序的先河,他们针对Cray X-MP计算机设计了共享内存的并行PIC程序ARGUS和CANDOR。该方法将计算域划分为若干子区域,每个进程的内存仅存储对应区域内的网格点值(包括电势、电场、数密度等)和该区域内的粒子信息。跨区域运动的粒子则被存储在CPU的缓存中,图6显示了一种一维的均匀区域分解方式。

在这种方法中,各个进程协同求解整个计算域的电场,再独立推动各自区域内的粒子并求解碰撞。当粒子运动跨过区域间的边界时,需要将该粒子的信息进行进程之间的转发。为了避免并行效率受到运行最慢的进程制约,还需要在运行时根据各进程的计算负担重新进行负载分配,本文称其为人工负载均衡,这部分将在4.3节中进行详细描述。

与串行程序相比,采用区域分解方法的并行程序主要需要改写以下部分:

(1)网格量与粒子存储的相关代码。

(2)电磁场求解的并行化(见4.2节)。

(3)负载均衡部分(见4.3节)。

(4)计算域间粒子信息通信。

图7显示了串行代码改为区域分解方法的代码流程,其中蓝色部分是涉及到通信的部分。

可以看出,区域分解算法主要有两种处理需要

Table 3 A summary of different kinds of parallelization methods

Object	Program model	Hardware	Granularity	Performance	Portability	Scalability
Vectorization	Vector	CPU	Fine	Low	Weak	Weak
Pthread	Threads	CPU	Coarse/Fine	Low	Weak	Weak
MPI	Message passing	CPU/GPU	Coarse/Fine	High	Strong	Strong
OpenMP	Shared memory	CPU/Cluster	Fine	High	Weak	Strong
CUDA	Shared memory	GPU	Fine	High	Weak	Strong

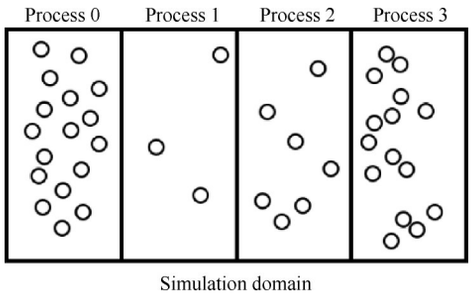

Fig. 6　One-dimensional uniform domain decomposition[54]

进行进程间通信：

（1）电磁场求解时，在每个子区域内部分别求解电磁场信息。但若某一子域的边界条件与其相邻子域中的电荷分布信息有关，则需要进行对应进程间的通信。这部分通信是相邻子域对应进程之间的一对一通信。

（2）在粒子推动之后，有一些粒子会从某一子域移动到另一子域中，这些跨越子域边界的粒子需要进行消息传递，将粒子的信息转移到新进程进行计算。这部分通信同样是相邻子域对应进程之间的一对一通信。

（3）另外，粒子跨越子域边界之后会导致不同进程负责的子域计算量不再一致，所以需要进行负载均衡步骤。这部分可能也涉及到区域之间的通信，如重新划分子域之后，老子域对应进程的粒子需要将信息传递给新子域对应的进程中。这部分涉及到的属于新老子域之间的一对一通信。

由上述分析可知，区域分解算法的通信代价主要是进程间一对一的通信，而非一对多或多对一的通信。同时，其通信量与网格数无关，而主要与子域数和粒子数有关。当需要模拟的问题规模变大时，区域分解算法主要关注粒子数的增长，而非网格数的增长。所以，该算法总通信量较小，适合大规模分布内存或混合并行方法。

由于区域分解的可拓展性强，在大规模集群中的性能上限高，采用这种分解手段的程序占到了绝大多数[9,12-13,15-17,21,23,46,56,67]，而粒子分配方法则使用较少。

粒子分配方法：纯粹的，或是“狭义”的粒子数均匀分配的方法首先由 Martino 等[112]提出，其主要的分解对象是粒子数而非区域。在“狭义”的粒子数均匀分配方法中，粒子被均匀的分配到各个进程中，每个进程拥有自己的粒子信息和网格参数（包括电势、电场、数密度等）。和区域分解算法不同的是，在该方法中，一个进程内的粒子可以分布在整个计算域的

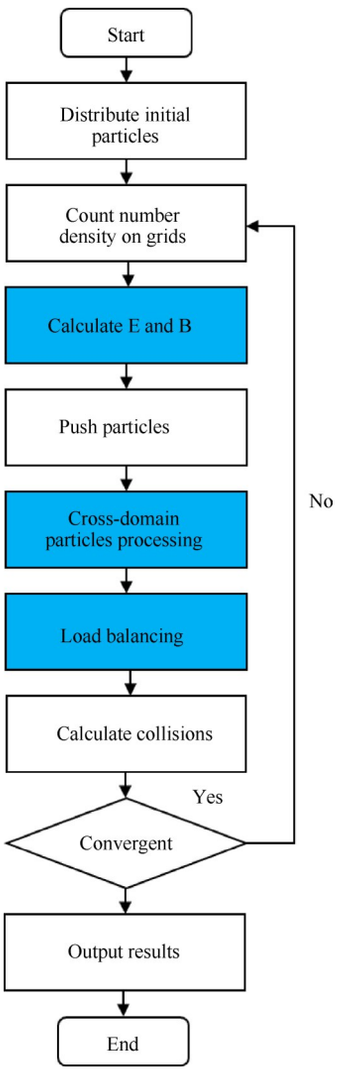

Fig. 7　Scheme of particle decomposition parallel method

任何位置，且一个进程内的网格量是整个计算域的网格量。简而言之，若存在 p 个进程，而粒子总数为 N 个，则每个进程处理 N/p 个粒子，各个进程单独实现此部分粒子的推动，而碰撞和电磁场信息部分则需要进程协同求解。其基本原理如图 8 所示。

在粒子数均匀分配方法中，当每个进程内的初始粒子的数量、位置、速度都大致相同时，则无论针对何种物理模型，经过多少运算时间，这些进程内粒

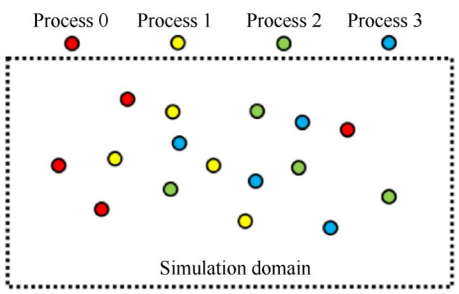

Fig. 8　Particle decomposition parallel method[54]

子的各项性质始终都将保持大致相同，本文称这种特性为天然负载均衡。

天然负载均衡的原因可以概括为以下两点：

（1）等离子体中的碰撞是随机的。对于不同进程中两个初始时刻粒子数相等、位置速度分布相近的粒子群，一段时间后因碰撞而损失或电离增加的粒子数大致相等。

（2）等离子体中粒子运动也是随机的。不同进程中两个初始条件相同的粒子群，在相同电磁场的推动下，经过随机碰撞后，运动情况大致相同。

与串行 PIC 程序相比，采用“狭义”的粒子数均匀分配方法的并行 PIC 程序主要有以下改动：

（1）新粒子的均匀分发算法；

（2）电磁场、碰撞处理以及粒子推动的前后处理中涉及通信；

（3）电磁场求解的并行化（见 4.2 节）；

（4）增加碰撞对选取函数，该函数涉及通信。

图 9 显示了串行代码改为狭义的粒子均匀分配的并行方法的代码流程，其中蓝色部分是涉及到通信的部分。

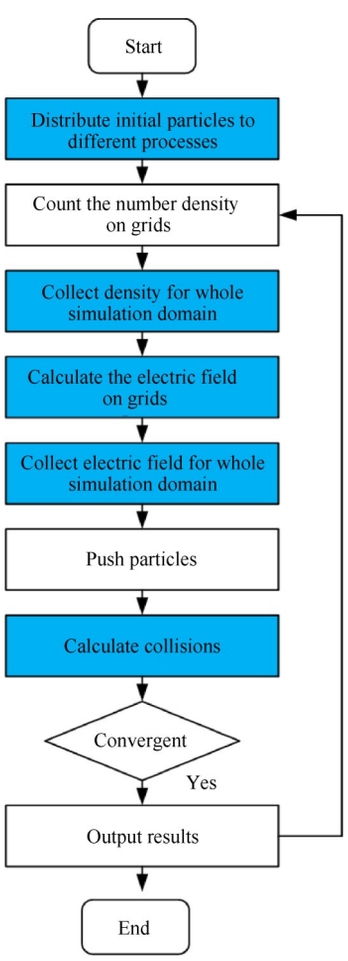

Fig. 9　Scheme of particle decomposition parallel method

可以看出，除了处理较为简单的粒子均匀分发算法外，狭义的粒子分配算法主要有两种处理需要进行进程间通信：

（1）电磁场处理部分：在求解电磁场时，程序需要将所有进程中的粒子数密度汇总到一个进程之后得到整个计算域的整体电荷密度，再求解域内的电磁场，此时会发生多对一通信。同时，程序需要将求解得到的电磁场信息再次分发给各进程，提供下一步迭代时推动各进程中粒子的电磁力，此时发生的是一对多通信。

（2）粒子的碰撞处理部分：由于粒子分散在整个计算域中，且无法预估粒子的位置，所以在选取碰撞对时，被碰撞粒子的位置和碰撞粒子可能不在同一进程中。程序只能遍历整个计算域中的所有粒子找到对应的碰撞对，此过程需要将所有进程中的粒子进行位置汇总到一个进程内，之后再进行遍历，发生的是多对一通信。

“狭义”的粒子数均匀分配方法的特点与区域分解方法正好相反：它的通信量与粒子数无关，而主要与网格数有关。由于每个进程都包含了整个计算域的物理参数和网格量，所以一对多和多对一是主要通信方式，通信耗时会随着计算规模的增大而急剧升高。

由于粒子数均匀分配方法对于串行 PIC 程序的改动较为简单，所以可以快速应用在在小规模的并行加速上，麻省理工学院的 Fox[12] 在其博士论文中就使用了该种方法，北京航空航天大学的并行 JLPP2.5 代码[53-54] 也使用了该种分解方案。

“广义”的粒子均匀分配算法：粒子的并发处理。举例说明：若采用 100 个进程模拟 10 个计算子域，则每个子域是由 10 个进程同时处理，此时一个子域内粒子的部分模拟（如推动和碰撞后处理）可被 10 个进程同时并发处理，但跨越子域的处理等依然是使用区域分解算法的原则。同样，同一子域对应的多个进程也可以使用 OpenMP 等手段对该子域内的电磁场进行并发求解。在实际处理中，只要计算域的子域数目少于处理器的进程数，程序都会加入这种朴素的粒子并发算法。

综上，对比区域分解和粒子分配（此处取“狭义”方法）两种方法，可以得到：

（1）在串行改并行的角度：区域分解法需要修改数据存储、负载均衡、粒子信息转发等部分；粒子分配法需要修改初始粒子分配、收集数密度、处理碰撞中目标粒子等部分。在修改难度方面，区域分解法

更难。

(2)从通信量角度来分析:区域分解法的通信量小,不同进程之间主要是和邻近的进程进行一对一通信;而粒子分配法由于每个进程都计算全局的电磁场信息和粒子信息,所以多对多通信量会随着规模的变大急剧增长。在通信量方面,区域分解方法更优。

(3)从可拓展性角度:随着计算进程的变多,计算的读入规模并没有变大,区域分解方法的通信量并未急剧增加,但粒子分配法中的通信量超线性增加,这将成为制约并行性能的主要障碍。所以区域分解方法在可拓展性方面有显著优势,在大型集群上粒子分配方法带来的加速比显著低于区域分解方法。

综上,可得一个PIC并行方法的选择准则:当并行计算的规模不太大,参与并行计算的进程数不太多时,程序可以使用天然负载均衡、编写较为简单的粒子分配方法(狭义);当计算规模很大,或者追求更高的性能上限时,应当选取通信量较小、可拓展性强的区域分解法。实际上,为了减少通信量,在多核处理器中,一般将子域数目小于总进程数,此时广义的粒子并发算法和区域分解算法相结合可以获得更高的计算效率。

4.2 电磁场求解算法的并行化

在电推进并行模拟中,考虑到通信时间,电磁场的求解耗时占据运行总时间的很大一部分。而电磁场的求解主要有三种方式:(a)求解准中性的玻尔兹曼方程[13,56],此方法准确度较差;(b)求解完整麦克斯韦方程组[67],此方法计算精度高但是计算代价大;(c)求解高斯定律或泊松方程(忽略磁场的变化)。在众多电推进装置中,一些电磁式推力器,如SF-MPD(自身场MPD推力器)、PPT(脉冲等离子体推力器)、PIT(脉冲诱导推力器)等需要考虑自身场,必须使用方法(b);而其他电推力器,如霍尔推力器、离子推力器等不需要考虑自身场,可以使用方法(c)。本文主要着眼于方法(c)的并行化研究进行介绍。

泊松方程离散后得到的是一个大型线性方程组,因此可使用求解线性方程组的一般迭代求解技术,如雅可比迭代、G-S迭代、超松弛迭代(Successive Over Relaxation,SOR)、共轭梯度法(Conjugate Gradient Method,CGM)等。若考虑到其作为椭圆方程的特性,则可应用交替隐式方向迭代(Alternating Direction Implicit,ADI)、动态交替隐式方向迭代(Dynamic Alternating Direction Implicit,DADI)、多重网格法(Multi-Grid Method,MGM)等进行求解。除上述的迭代求解方法,还有若干直接方法,如循环约化法(Cyclic Reduction,CR)、快速傅里叶变换(Fast Fourier Transform Method,FFT)等。华中科技大学的姜巍[113]对上述方法在速度、通用性和并行难易度方面作了比较,见表4。

根据电推进装置的特点,本文选取了常用泊松方程求解的具体方案:

(1)快速傅里叶变换(FFT)

FFT方法被广泛用于并行静电和电磁PIC代码中,并在多个研究都证明了其高效的并行效率。JASMIN框架设计了基于OpenMP的并行FFT并行求解器[98-99];加州理工大学洛杉矶分校的UPIC[21]系列代码使用了并行化的FFT方法,并在其他领域的粒子模拟中得到了应用[114-115]。但是,大多数电推进装置的内部边界都较为复杂,由于泊松方程的椭圆性质,无法应用局部非迭代场求解方法。所以在复杂边界的电推进粒子模拟中,FFT方法适用性差。

(2)超松弛算法(SOR)

SOR算法是一种常用的求解五点差分格式下高斯定律的方法[116],JASMIN框架开发了基于OpenMP和MPI混合的并行SOR求解器[98]。Fox[12]在将MIT-PIC代码进行并行化时,使用了并行的红黑SOR算法[117],该方法将节点分组,使得矩阵方程解耦且更易

Table 4 Comparison of different electric field solving methods[113]

	Methods	Speed	Universality	Parallelization
Direct method	Cyclic reduction (CR)	Very fast	Bad	Hard
	Eigenvalue decomposition	Slow	Very bad	Easy
	Fast Fourier transform method (FFT)	Very fast	Very bad	Very easy
Iteration method	Successive over-relaxation (SOR)	Very slow	Better	Easy
	Multi grid	Fast	Good	It depends
	Alternative direction implicit (ADI)	Fast	Better	Very hard
	Conjugate gradient	Fast	Good	Hard
	Generalized minimal residual method	Fast	Best	Very hard

于并行化，如图10所示，其基本的算法是：(a)每个处理器接收一条节点，(b)首先计算节点的第一行(黑色节点)，(c)之后按照从上至下、从左至右的顺序，计算其余的“红色”节点。

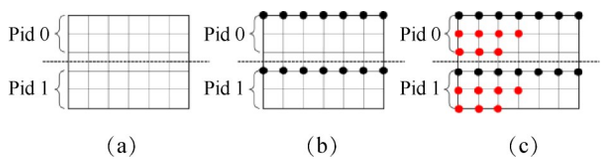

Fig. 10　Paralleled red-black SOR algorithm

(3)动态交替方向隐式算法(DADI)

动态交替隐式方向迭代算法(DADI)算法是由ADI发展而来的，ADI的主要算法是通过引入虚拟时间项$\partial\phi/\partial t$(ϕ是电势)，将求解电势的过程等价于求解一个热传导问题的稳态解。之后引入一个虚拟迭代时间步长Δt_f，不断进行行列迭代，算法如图11所示。而DADI是在ADI算法的基础上，将Δt_f设置成一个可变值，逐渐迭代得到收敛的电势数值解。

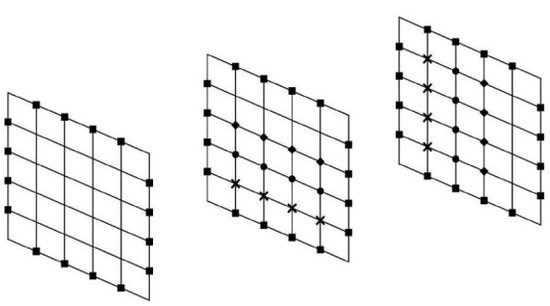

Fig. 11　ADI algorithm with *x*- and *y*-direction iteration

虽然DADI的并行化相比于FFT等方法较难，但DADI的行迭代过程中各行方程无关，在列迭代过程中各列方程也无关，所以求解过程可以由各线程分别进行，从而实现并行加速，如图12所示。

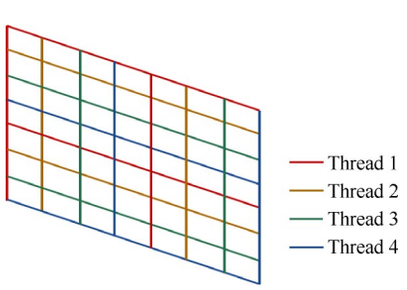

Fig. 12　Paralleled DADI algorithm

Mahalingam的离子推力器仿真代码[9,42]，北京航空航天大学的并行化JLPP2.5[53-54]代码和通用并行化框架GPIC[73]都使用了该方法。

(4)PETSc工具包

PETSc是由美国阿贡国家实验室(Argonne National Laboratory，ANL)开发的一套开源的、便携式的、可扩展的科学计算工具包。该工具包是一套数据结构和例程，主要用于科学计算中偏微分方程、线性方程组等的快速求解。它是完全并行化的，支持MPI以及GPU加速手段(CUDA、OpenCL)，同时支持混合MPI-GPU混合并行加速。日本航天局JAXA的Cho等[16]在其并行代码中，使用PETSc进行电磁场求解加速；西安交通大学李昊霖等[64,118]在对栅极离子引出机制的研究中也通过PETSc进行了线性方程求解的并行加速。在非电推进领域，PETSc也被应用于托克马克的BOUT++程序中[119]。

(5)其他方法

弗吉尼亚理工大学的Pierru[27]开发了适用于浸入式有限元的子域并行IFE场求解器，其本质是改进的SOR算法，该方法同样被哈尔滨工业大学(深圳)的曹勇课题组使用[62]；Ortwein等[120]开发了一种不连续伽辽金谱元法的电磁场并行求解器，被用在PIC-DSMC代码中的的电磁场求解部分；CHAOS框架[46]使用的是改进的预处理共轭梯度法(preconditioned conjugate gradient method)；北京航空航天大学的Ren等[51]和仇钎[52]在GPU并行化的离子推力器模拟中使用了DADI和代数多重网格法(Algebraic Multi Grid method，AMG)的复合电磁场求解方法。

4.3　并行算法中的动态负载均衡

负载均衡问题对于并行化系统是一个重要的命题。由3.1节可知，粒子分配的算法具有天然负载均衡性，但在使用区域分解方法时，如何保证在计算中各进程中粒子数和网格数都大致相同是影响计算性能的一大重要因素。一些研究人员在其代码中没有做动态负载均衡的尝试[9,13,56]，这将显著降低代码的效率。下面介绍三大类对于负载平衡所做的研究，其中一些并不直接应用在电推进领域，但是其基本思想是值得借鉴的。

第一大类的思路是动态的重新划分计算域，使得每个计算域内的粒子数量大致相等。其基本原理如图13所示。

Ferraro等[121]在1993年为在MIMD计算机上的GCPIC代码实施了负载平衡方案：每当任何处理器的粒子计数超过某个不平衡阈值时，都会通过重新调整GCPIC主分区来维持粒子负载平衡。研究者通过建立一个网格沉积电荷分布计算的一维数密度函数，在一维方向上来确定新的分区边界。该算法

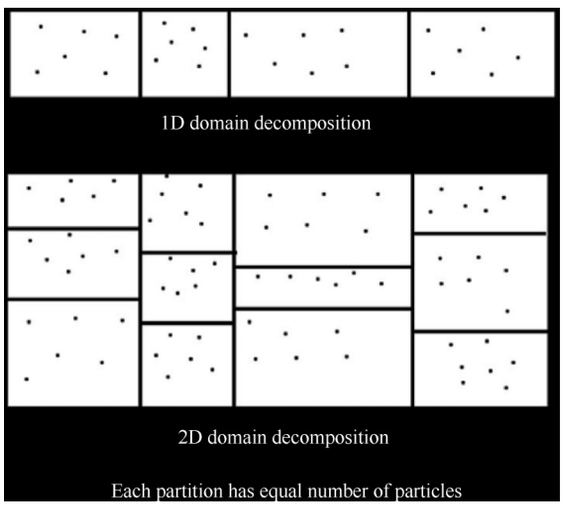

Fig. 13　Diagram shows domain decomposition among different processors in a parallel PIC code[92]

已被应用于使用粒子和场的一维分区的二维静电等离子体 PIC 仿真，并且已在 Caltech/JPL Mark III，NCUBE2 和 Intel i860 Gamma 超立方体计算机上运行。测试发现：负载平衡后粒子推动的并行效率非常高（90%~100%），并且随着处理器数量的增加体现出很好的可拓展性。

由法国的 Derouillat 等[122]开发的 SMILEI 是一种基于 C++和 Python2.7 语言的、多用途的、面向对象开发的三维等离子体仿真框架。SMILEI 框架采用了 MPI 的手段，在区域分解中，采用希尔伯特曲线来进行区域划分和动态负载均衡。本文将简单介绍其基本的算法。

图 14 给出了一个希尔伯特曲线将计算域划分为子域的一个例子。当计算域为 32×32 区块，进程为 7 个时，采用希尔伯特曲线（黑线）穿过所有的区块中心（黑点），从坐标为（0，0）的区块中心点开始，到坐标为（31，0）的区块中心点结束。曲线被分成与 MPI 进程一样多的分段，每一段覆盖一个子域，由对应的进程处理。不同的子域用不同颜色表示。

算法中的动态负载的主要思路为：当某个 MPI 进程过载时，对应的希尔伯特曲线分段向其相邻分段发送自身管辖的网格，使得该分段变短；相反的，一个负载不足的进程将从其相邻分段接收计算网格，其曲线片段变长。其具体的平衡算法如下：首先，每个子域 p 的计算负载大小被评估为 $L_p = N_{\mathrm{part}} + C_{\mathrm{cell}} N_{\mathrm{cell}} + C_{\mathrm{frozen}} N_{\mathrm{frozen}}$。其中 N_{part} 是子域中的活动粒子数，N_{cell} 是子域中的网格数，N_{frozen} 是块网格冻结（不动）粒子的数量，C_{cell} 和 C_{frozen} 是用户定义的系数，表示网格和冻结粒子的计算成本。整个计算域的总

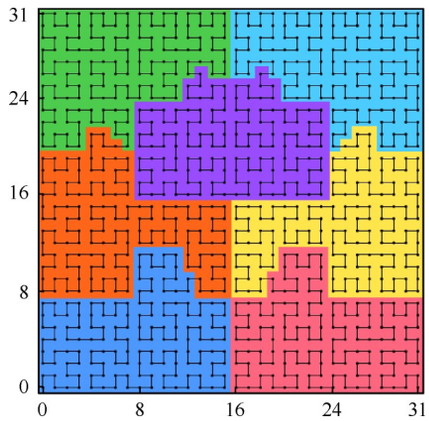

Fig. 14　32 × 32 grids decomposition for 7 processors using Hilbert curves[122]

负载为 $L_{\mathrm{tot}} = \sum_p L_p$，每个进程的最佳计算负载 $L_{\mathrm{opt}} = L_{\mathrm{tot}}/N_{\mathrm{MPI}}$，其中 N_{MPI} 是 MPI 进程的数量。在大多数情况下，活动粒子是计算负荷的主要来源，算法会根据 L_p 与 L_{opt} 的大小关系对希尔伯特曲线进行新的分解，以便每段的载荷尽可能接近 L_{opt}。此过程间隔一段时间就重复一次，以保持各个进程计算负载的平衡。图 15 显示了一个二维算例中，MPI 区域分割情况随计算负载分布的演化，上方的色卡表示的是局部不平衡度 $I_{\mathrm{loc}} = \lg(L_{\mathrm{loc}}/L_{\mathrm{av}})$，越接近于 1 表示越平衡，其中 L_{loc} 为局部块计算负载，L_{av} 为平均计算负载。黑色线条划定了不同的 MPI 域，激光等离子体从计算域左侧进入并向右传播，4 个图片从上到下依次显示了经过 1600，5480，6820，9080 次迭代后的整个仿真域。可以看出经过动态划分计算域后，整个计算域内的负载不均衡度显著下降。

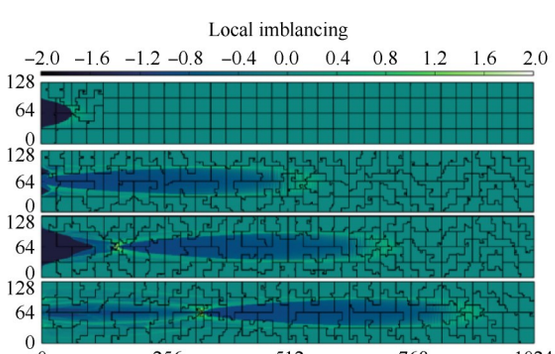

Fig. 15　Dynamic loading when the plasma density changes using Hilbert curves[122]

基于 Metis 库的图划分方法，也采用同样的思路进行动态划分计算域。韩国世宗大学的 Lee[17]和国防科技大学的张海红[67]在 DSMC 仿真时，都使用 Metis 库来进行动态负载均衡。

第二大类的思想就是通过将繁忙进程中的粒子经过通信分配到空闲进程，以实现各个进程的负载近似相同。Othmer等[123]在2002年提出了一种名为Taskfarm的动态负载平衡策略。其基本算法如下：首先，在一维方向上进行均分的区域划分，不同区域中的粒子计算在所有的计算进程之间进行交易(Trade)。当一个进程完成了本区域内粒子的计算工作，就会将其他的进程中的工作抓取过来，处理后再将处理之后的信息写回原来的计算进程。在初始并行分区之后，一个主进程被赋予了“任务主管”的额外职责，通过一个任务列表可以将任务分配给其他的进程。剩余的进程是从属进程，在处理完本身的工作之后，那些从属进程会收到计算其他进程或是终止迭代的任务。如果从属进程收到了接受任务的指令，则需要通过通信获得高负载进程中的粒子和电磁场信息，将这些数据复制到自己的本地存储器后开始计算，计算完成后再将信息存放回原始目标内存中。算法中还引入了自锁结构防止同一部分的内存被重复的访问，并且只有本进程所有的任务都完成之后才可以接受其他进程的任务。图16比较了16进程下，一个一维区域分解算例在使用Taskfarm算法前后，各进程中的粒子数随迭代时间步(横坐标为Δt)的变化情况。该算例模拟了一束激光等离子体从计算域左侧沿x方向运动到计算域右侧。图16中的上图显示了无Taskfarm算法的情况：当等离子体到达一个子域后，对应进程中的粒子数开始上涨直到饱和，不同子域对应的各进程重复上述过程，导致同一时刻时各进程负载不均。图16中的下图则显示了使用Taskfarm算法进行负载均衡后的负载情况：随着迭代的进行，每个进程中的粒子数随时间变化的趋势近似相同，即负载均衡性得到大幅提高。

日本神户大学的Usui等[124]开发的OhHelp是一种基于区域分解的、可拓展的、负载平衡的PIC模拟算法。这套算法使用了一种“互帮互助”式的负载平衡机制。在计算过程中空闲进程会帮助高负载进程，负高负载进程在对应子域中配置了一部分粒子，并将粒子信息复制给其助手(Helper)。程序中设计了一个负载平衡检测器，用于检测粒子的跨边界移动和各进程间负载是否均衡，当检测到不可接受的不平衡时重新配置计算负载。其基本的算法是在初始阶段均匀划分计算区域到每个计算进程，之后定义一个非均匀函数，当每个进程上的粒子数目P_{d}满足$P_{\mathrm{d}} \leqslant (P/N)(1+\alpha) \equiv P_{\mathrm{lim}}$时，即记为非均匀状态。其中$P$是总粒子数目，$N$是进程数目，$\alpha$是非均匀容忍

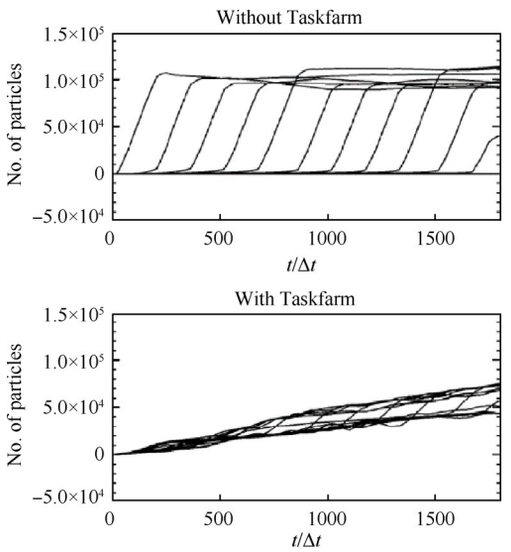

Fig. 16　Particle numbers versus time, with and without Taskfarm algorithm[123]

度。当首次划分计算区域和进程后，对每个进程的粒子数目进行统计，若不均匀则将高负载进程中过多的粒子分配给其他进程计算，直到每个进程的粒子数目都接近P_{lim}。图17显示了16进程下，一个二维区域分解算例使用OhHelp算法进行负载均衡前后各进程中的粒子数目。图17(a)显示了未使用负载均衡前，各个进程中粒子数目都偏离于各进程的平均粒子数(Average number of particles)，图17(b)则显示经过OhHelp算法进行负载均衡之后，各个进程中的粒子数目都可以与平均粒子数持平，负载均衡性得到大幅提升。

第三大类是通过改变粒子权重，动态地将每个进程中的宏粒子数目保持大致相同。Assous等[125]提

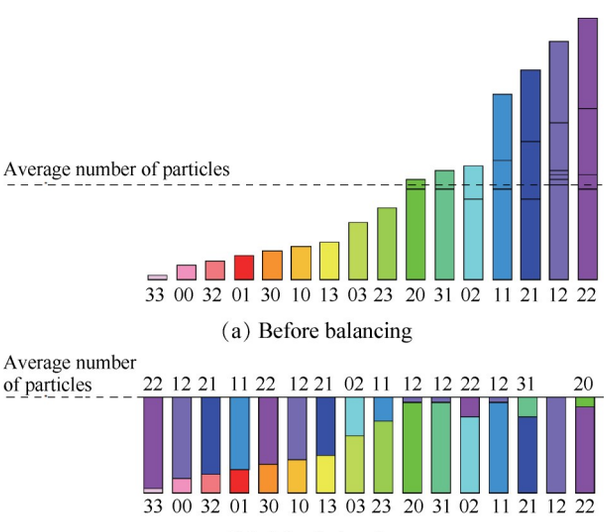

Fig. 17　Dynamic loading among 16 computational nodes using OhHelp [124]

出了控制宏粒子数的方法来保持PIC代码中空间宏粒子数的均匀性，Teunissen等[126]使用了k-d树（k-d tree）的数据结构，对PIC代码中的粒子权重进行了动态重新赋值的操作，当一子域内的粒子数较多时，增大粒子权重，使得单个宏粒子代表更多真实粒子；反之，若一子域中的粒子较少，则采用较小粒子权重。为保持每个网格内的宏粒子要大于一定数值以满足粒子方法的统计性原理，常使用粒子合并或粒子分裂的方式来保持宏粒子数目保持一定数量。西安交通大学的孙安邦课题组[64,118]的栅极离子引出代码中就使用了变粒子权重的方法来实现动态负载均衡。

当然，也存在多种方法相结合的算法，如CHAOS框架[46]实现动态负载均衡时就同时使用了动态网格调节/八叉树（AMR/Octree）技术和进程间通信技术。八叉树算法是一种三维的、用来储存不同节点中的粒子的多层树形数据结构，不同节点实际代表了一个特定的计算区域。整个计算区域是树结构的根节点（Root node），之后进行迭代划分，此时子域成为叶节点（Leaf node）。经过上述处理后，计算域便形成了一个八叉树森林（Forest of octrees，FOT），可针对不同的问题规模进行拓展。这样的FOT可以有多种，CHAOS在模拟中就设置了两套FOT[127]：对于DSMC模拟，节点会持续迭代直到最后的叶节点中满足平均自由程条件，最终形成C-FOT；而对于PIC算法中的电磁场求解，最后的叶节点则需要满足德拜长度的条件，形成E-FOT。在实际分解中，CHAOS先采用多进程平均划分计算域到若干子域，再使用2∶1原则对子域进行动态网格划分[128]，形成两套FOT网格。在二维情况下，八叉树会退化成四叉树，图18展示了四叉树算法的动态多级网格划分算法。

在实现并行网格动态划分之后，每个网格内的粒子数目和计算负载接近，但若各子域内所含网格数不同，进程之间会发生负载不均衡。此时，CHAOS通过对每个实际网格进行权重赋值（一般都为1），来判断各个进程的实际负载量，再通过进程间通信实现负载均衡。举例说明：若计算域被分为了两个子域，分别由进程p_0和进程p_1计算。经过动态网格划分后，进程p_0有30个权重为1的网格，而进程p_0只有20个，此时进程p_0会将5个网格中的粒子信息传递给进程p_1进行计算，实现负载均衡。CHAOS的优势在于其网格大小是自适应变化的，避免了过多网格的生成，在兼顾动态负载的同时，降低了算例的总体计算量。

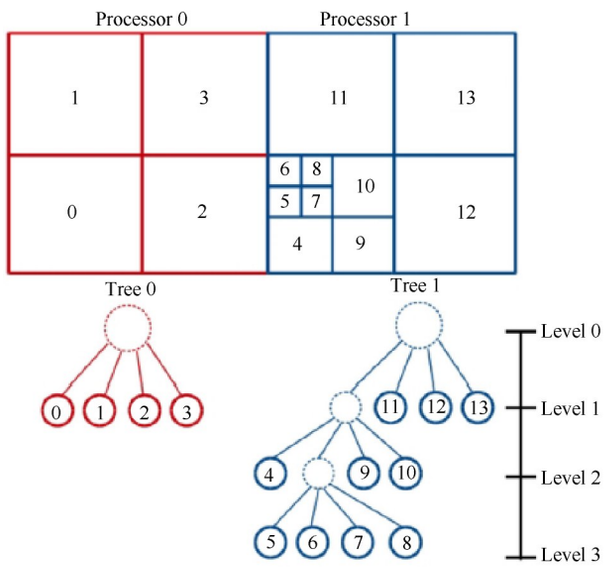

Fig. 18　AMR/Octree algorithm of CHAOS

总结前述的三大类算法，动态改变计算域可以在通信量较少的情况下达到很好的负载效果，但是无论是希尔伯特曲线还是八叉树算法，代码书写都有一定难度；进程之间相互“帮助”以实现代码均衡的算法不可避免地带来了通行量的提升，而进程间的通信会拖慢计算时间；改变权重的办法虽然简单，但是改变了正常情况下的粒子总数目，在一定程度上真实性不如前两种算法。

5　总结和展望

电推进粒子算法物理假设少但是计算量大，并行化计算研究不可或缺。经过多年发展，国外的并行化技术和框架研究已经取得了丰富的成果，而国内相关研究由于起步较晚还处于高速发展阶段。本文针对并行算法的框架设计和算法实现的梳理，得到了如下的一些主要结论和展望：

（1）现阶段，粒子算法并行研究的主流集中在面向对象的通用并行框架设计上，以期实现界面化、模块化、通用化，让研究者以较小的代价实现更多的计算可能性。

（2）粒子算法的并行研究正朝着超大集群、混合加速手段（CPU/GPU）的方向上发展，随着以GPU为代表的计算单元算力的进一步提升，计算效率的上限值将被进一步提升。

（3）随着AI专属异构架构的逐渐成熟，通用异构未来可能与AI专用异构计算实现融合。多种架构的并行方式也可能逐渐实现无缝移植、统一编程、一致存算等功能。

(4)其他领域,如核聚变、激光等离子体、等离子体刻蚀、空间等离子体等,也同时使用粒子算法进行等离子体模拟。电推进领域的粒子模拟相比于上述领域而言,起步较晚,研究人员较少,所以可以从其他领域内吸收优秀算法,如动态负载算法、电磁求解算法等。

(5)对于国内研究人员,大多粒子代码处于串行改并行阶段,且面向对象设计并不完全,如何实现快速串改行是主要任务,所以应当先从简单的手段入手:如使用OpenMP进行自动优化,采用粒子均分的分解算法避免动态负载代码,采用PETSc等成熟工具包来实现电磁场的加速等。

参考文献

[1] 康小录,张 岩. 空间电推进技术应用现状与发展趋势[J]. 上海航天, 2019, 36(6): 24-34.

[2] 耿 海,李 婧,吴辰宸,等. 空间电推进技术发展及应用展望[J]. 气体物理, 2022, *DOI*: 10.19527/*j.cnki.*2096-1642.0977.

[3] 耿 海,吴辰宸,孙新锋,等. 高功率空间电推进技术发展研究[J]. 真空与低温, 2022, 28(1): 14-25.

[4] 康小录,张 岩,刘 佳,等. 大功率霍尔电推进研究现状与关键技术[J]. 推进技术, 2019, 40(1): 1-11. (KANG Xiao-lu, ZHANG Yan, LIU Jia, et al. Research Status and Key Technologies of High-Power Hall Electric Propulsion[J]. *Journal of Propulsion Technology*, 2019, 40(1): 1-11.)

[5] 李 永,周 成,吕 征,等. 大功率空间核电推进技术研究进展[J]. 推进技术, 2020, 41(1): 12-27. (LI Yong, ZHOU Cheng, LYU Zheng, et al. Progress on High Power Space Nuclear Electric Propulsion Technology Development [J]. *Journal of Propulsion Technology*, 2020, 41(1): 12-27.)

[6] Verboncoeur J P. Particle Simulation of Plasmas: Review and Advances[J]. *Plasma Physics and Controlled Fusion*, 2005, 47(5*A*): *A*231-*A*260.

[7] Szabo J J. Fully Kinetic Numerical Modeling of a Plasma Thruster[D]. *Cambridge*: *Massachusetts Institute of Technology*, 2001.

[8] Hirakawa M. Electron Transport Mechanism in a Hall Thruster[C]. *Cleveland*: 26*th International Electric Propulsion Conference*, 1997.

[9] Mahalingam S. Particle-Based Plasma Simulations for an Ion Engine Discharge Chamber[D]. *Dayton*: *Wright University*, 2010.

[10] Taccogna F, Longo S, Capitelli M, et al. Self-Similarity in Hall Plasma Discharges: Applications to Particle Models[J]. *Physics of Plasmas*, 2005, 12(5): 053502.

[11] 张林波. 并行计算导论[M]. 北京: 清华大学出版社, 2006.

[12] Fox J M. Parallelization of Particle-in-Cell Simulation Modeling Hall-Effect Thrusters[D]. *Cambridge*: *Massachusetts Institute of Technology*, 2005.

[13] Roy R I S, Hastings D E, Taylor S. Three-Dimensional Plasma Particle-in-Cell Calculations of Ion Thruster Backflow Contamination [J]. *Journal of Computational Physics*, 1996, 128(1): 6-18.

[14] Cai C, Boyd I. 3D Simulation of Plume Flows from a Cluster of Plasma Thrusters [C]. *Toronto*: 36*th AIAA Plasmadynamics and Lasers Conference*, 2005.

[15] Fergason S. Ion Thruster Plume Simulation Using Clustered PC Workstations[C]. *Reno*: 43*rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

[16] Cho S, Komurasaki K, Arakawa Y. Kinetic Particle Simulation of Discharge and Wall Erosion of a Hall Thruster [J]. *Physics of Plasmas*, 2013, 20(6): 063501.

[17] Lee K H. Multi-Plume Flow Simulation of Small Bipropellant Thrusters Using Parallel DSMC Method[J]. *Computers & Fluids*, 2018, 173: 259-263.

[18] Liewer P C, Decyk V K. A General Concurrent Algorithm for Plasma Particle-in-Cell Simulation Codes[J]. *Journal of Computational Physics*, 1989, 85(2): 302-322.

[19] Decyk V K. Skeleton PIC Codes for Parallel Computers [J]. *Computer Physics Communications*, 1995, 87(1-2): 87-94.

[20] Norton C D, Szymanski B K, Decyk V K. Object-Oriented Parallel Computation for Plasma Simulation[J]. *Communications of the ACM*, 1995, 38(10): 88-100.

[21] Decyk V K. UPIC: A Framework for Massively Parallel Particle-in-Cell Codes[J]. *Computer Physics Communications*, 2007, 177(1): 95-97.

[22] Huang C, Decyk V K, Ren C, et al. QUICKPIC: A Highly Efficient Particle-in-Cell Code for Modeling Wakefield Acceleration in Plasmas[J]. *Journal of Computational Physics*, 2006, 217(2): 658-679.

[23] Wang J, Liewer P, Decyk V. 3D Electromagnetic Plasma Particle Simulations on a MIMD Parallel Computer [J]. *Computer Physics Communications*, 1995, 87(1): 35-53.

[24] Wang J, Cao Y, Kafafy R, et al. Simulations of Ion Thruster Plume Spacecraft Interactions on Parallel Supercomputer [J]. *IEEE Transactions on Plasma Science*, 2006, 34(5): 2148-2158.

[25] Kafafy R, Yong C, Wang J. Electric Propulsion Plume Interactions with Formation Flying Spacecraft[C]. *Florence*: 30*th International Electric Propulsion Conference*, 2007.

[26] Wang J, Cao Y, Kafafy R, et al. Electric Propulsion Plume Simulations Using Parallel Computer[J]. *Scientific Programming*, 2007, 15(2): 83 - 94.

[27] Pierru J. Development of A Parallel Electrostatic PIC Code for Modeling Electric Propulsion[D]. *Blacksburg*: *Virginia Polytechnic Institute and State University*, 2005.

[28] Brieda L. Development of the DRACO ES-PIC Code and Fully-Kinetic Simulation of Ion Beam Neutralization[D]. *Blacksburg*: *Virginia Polytechnic Institute and State University*, 2005.

[29] Spicer R L. Validation of the DRACO Particle-in-Cell Code Using Busek 200W Hall Thruster Experimental Data[D]. *Blacksburg*: *Virginia Polytechnic Institute and State University*, 2007.

[30] Hu Y, Wang J. Electron Properties in Collisionless Mesothermal Plasma Expansion: Fully Kinetic Simulations [J]. *IEEE Transactions on Plasma Science*, 2015, 43(9): 2832-2838.

[31] Hu Y, Wang J. Fully Kinetic Simulations of Collisionless, Mesothermal Plasma Emission: Macroscopic Plume Structure and Microscopic Electron Characteristics[J]. *Physics of Plasmas*, 2017, 24(3): 033510.

[32] Verboncoeur J P, Langdon A B, Gladd N T. An Object-Oriented Electromagnetic PIC Code[J]. *Computer Physics Communications*, 1995, 87(1): 199-211.

[33] Mardahl P J, Verboncoeur J P. Progress in Parallelizing XOOPIC[C]. *Raleigh*: 25*th IEEE International Conference on Plasma Science*, 1998.

[34] Proulx N D, Jugroot M. Simulation of an Adaptable Hall Thruster[C]. *Vienna*: 36*th International Electric Propulsion Conference*, 2019.

[35] Kronhaus I, Hamo O. 2D Particle-in-Cell Channel Simulation of the Narrow Channel Hall Thruster[C]. *Vienna*: 36*th International Electric Propulsion Conference*, 2019.

[36] Jonell M, Menart J, Mahalingam S. Particle-Based Plasma Simulation of NEXT Ion Engine[C]. *Denver*: 45*th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2009.

[37] Lafleur T, Rafalskyi D, Aanesland A. Radio-Frequency Biasing of Ion Thruster Grids[C]. *Vienna*: 36*th International Electric Propulsion Conference*, 2019.

[38] Habl L, Rafalskyi D, Lafleur T. Ion Beam Diagnostic for the Assessment of Miniaturized Electric Propulsion Systems[J]. *Review of Scientific Instruments*, 2020, 91(9): 093501.

[39] Musso I, Manente M, Carlsson J, et al. 2D OOPIC Simulations of the Helicon Double Layer[C]. *Florence*: 30*th International Electric Propulsion Conference*, 2007.

[40] Mahalingam S, Menart J. Computational Model Tracking Primary Electrons, Secondary Electrons and Ions in the Discharge Chamber of an Ion Engine[C]. *Tucson*: 41*st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2005.

[41] Mahalingam S, Menart J A. Computational Study of Primary Electron Confinement by Magnetic Fields in the Discharge Chamber of an Ion Engine[J]. *Journal of Propulsion and Power*, 2007, 23(1): 69-72.

[42] Mahalingam S, Choi Y, Stoltz P, et al. Computational Speed up Techniques for Particle-in-Cell-Monte Carlo Collision Simulations of an Ion Engine Discharge Chamber[C]. *San Francisco*: 2013 *Abstracts IEEE International Conference on Plasma Science*(*ICOPS*), 2013.

[43] Nieter C, Cary J R. VORPAL: A Versatile Plasma Simulation Code [J]. *Journal of Computational Physics*, 2004, 196(2): 448-473.

[44] Ryan A, Bilek M, Cairns I H, et al. Magnetized and Unmagnetized Axisymmetric Particle-in-Cell Simulations of Ion Energy Distributions in Cathodic Vacuum Arcs[J]. *Plasma Sources Science and Technology*, 2022, 31(8): 085003.

[45] Brandt T, Trottenberg T, Groll R, et al. Simulations on the Influence of the Spatial Distribution of Source Electrons on the Plasma in a Cusped-Field Thruster[J]. *The European Physical Journal D*, 2015, 69(6).

[46] Jambunathan R, Levin D A. CHAOS: An Octree-Based PIC-DSMC Code for Modeling of Electron Kinetic Properties in a Plasma Plume Using MPI-CUDA Parallelization [J]. *Journal of Computational Physics*, 2018, 373: 571-604.

[47] Jambunathan R, Levin D A. Kinetic, 3-D, PIC-DSMC Simulations of Ion Thruster Plumes and the Backflow Region[J]. *IEEE Transactions on Plasma Science*, 2020, 48(6): 2017-2034.

[48] Averkin S N, Gatsonis N A. A Parallel Electrostatic Particle-in-Cell Method on Unstructured Tetrahedral Grids for Large-Scale Bounded Collisionless Plasma Simulations[J]. *Journal of Computational Physics*, 2018, 363: 178-199.

[49] Allison D, Baldwin J, Scharfe M. Development of BEPPA: An Object-Oriented Parallel Code for Full 3-D Spacecraft Plume Analysis and Satellite Design [C]. *Hartford*: 44*th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2008.

[50] Kühn C, Groll R. picFoam: An OpenFOAM Based Electrostatic Particle-in-Cell Solver [J]. *Computer Physics Communications*, 2021, 262: 107853.

[51] Ren J, Li J, Xie K, et al. Three Dimensional Simulation of Ion Thruster Plume-Spacecraft Interaction Based on a Graphic Processor Unit[J]. *Plasma Science and Technol-*

ogy, 2013, 15(7): 702 - 709.

[52] 仇 钎. 离子发动机交换电荷离子分布的数值模拟[D]. 北京: 北京航空航天大学, 2010.

[53] 李卓恒. 小功率霍尔推力器的PIC并行计算[D]. 北京: 北京航空航天大学, 2019.

[54] Pan R, Li Z, Cao S, et al. Parallel Codes Using Particles Decomposition and View Factor Model Methods for the Particle in Cell-Monte Carlo Collision (PIC-MCC) Simulation on Cylinder Hall Thruster[C]. *Vienna*: 36*th International Electric Propulsion Conference*, 2019.

[55] 蔡国飙, 刘世俭, 王慧玉, 等. 真空羽流场的DSMC并行数值模拟[J]. 航空动力学报, 1999, 14(2): 2-7.

[56] 王虹玥. 电推力器羽流的粒子网格法数值模拟[D]. 北京: 北京航空航天大学, 2010.

[57] 王虹玥, 贺碧蛟, 蔡国飙. 稳态等离子体推力器羽流仿真及其对微波的衰减和相移作用[J]. 推进技术, 2011, 32(6): 839-844. (WANG Hong-yue, HE Bi-jiao, CAI Guo-biao. Numerical Simulation of the Stationary Plasma Thruster Plume and Its Effects on the Microwave Attenuation and Phase Shifts[J]. *Journal of Propulsion Technology*, 2011, 32(6): 839-844.)

[58] Cai G, Zheng H, Liu L, et al. Three-Dimensional Particle Simulation of Ion Thruster Plume Impingement[J]. *Acta Astronautica*, 2018, 151: 645-654.

[59] Liu W, Cai G, Zhang J, et al. Numerical Investigation of Plasma Behavior in a Micro DC Ion Thruster Using the Particle-in-Cell/Monte Carlo Collision (PIC/MCC) Method[J]. *Journal of Physics D: Applied Physics*, 2021, 54(44): 445202.

[60] Hu Y, Wang J, Sun Q. Geometrically Self-Similar Ion Acceleration in Collisionless Plasma Beam Expansion[J]. *Plasma Sources Science and Technology*, 2020, 29(12): 125004.

[61] Hu Y, Huang Z, Cao Y, et al. Kinetic Insights into Thrust Generation and Electron Transport in a Magnetic Nozzle[J]. *Plasma Sources Science and Technology*, 2021, 30(7): 075006.

[62] Shan K, Chu Y, Li Q, et al. Numerical Simulation of Interaction Between Hall Thruster CEX Ions and SMART-1 Spacecraft[J]. *Mathematical Problems in Engineering*, 2015(3): 1-8.

[63] 白进纬. 二维隐格式IFE-PIC等离子体数值模拟及其应用[D]. 深圳: 哈尔滨工业大学, 2021.

[64] Li H L, Sun A B. Issues in the Numerical Modelling of Positive Ion Extraction[J]. *Computer Physics Communications*, 2021, 259: 107629.

[65] Sun A B, Becker M M, Loffhagen D. PIC/MCC Simulation of Capacitively Coupled Discharges: Effect of Particle Management and Integration[J]. *Computer Physics Communications*, 2016, 206: 35-44.

[66] Fu Y, Yang J, Jin Y, et al. Equivalent Two-Dimensional Numerical Simulation of an ECR Neutralizer[J]. *Acta Astronautica*, 2019, 164: 387-392.

[67] 张海红. DSMC/PIC等离子体羽流的大规模并行计算[D]. 长沙: 国防科技大学, 2020.

[68] 张志远. 稳态等离子体推力器三维羽流流场及其污染效应研究[D]. 长沙: 国防科技大学, 2015.

[69] Jin X, Huang T, Chen W, et al. GPU-Accelerated PIC/MCC Simulation of Laser-Plasma Interaction Using BUMBLEBEE[C]. *Savannah*: 57*th Annual Meeting of the APS Division of Plasma Physic*, 2015.

[70] 唐茂文. 1D3V粒子模拟软件BUMBLEBEE的GPU并行研究[D]. 成都: 电子科技大学, 2016.

[71] 郭胜龙, 金晓林, 杨明娟, 等. 离子推进器放电室电离特性的数值模拟研究[C]. 厦门: 真空电子学分会第二十届学术年会, 2016.

[72] Jin X, Huang T, Yang M, et al. A 2D3V Electromagnetic PIC/MCC Simulation of Plume Properties in Ion Thruster[J]. *IEEE Transactions on Plasma Science*, 2018, 46(8): 2814-2818.

[73] 毛仁凡. 空心阴极等离子体分布的并行模拟[D]. 北京: 北京航空航天大学, 2021.

[74] 潘若剑, 毛仁凡, 任军学, 等. 大功率霍尔推进器中阴极位置影响的粒子仿真研究[C]. 西安: 第18届中国电推进学术研讨会, 2022.

[75] Pan R, Ren J, Tang H, et al. Application of the View Factor Model on the Particle-in-Cell and Monte Carlo Collision Code[J]. *Physical Review E*, 2020, 102(3): 033311.

[76] Dawson J. One-Dimensional Plasma Model[J]. *The Physics of Fluids*, 1962, 5(4): 445-459.

[77] Dawson J M. Thermal Relaxation in a One-Species, On-Dimensional Plasma[J]. *The Physics of Fluids*, 1964, 7(3): 419-425.

[78] Goplen B, Ludeking L, Smithe D. Magic User's Manual[R]. *ADA*321106, 2016.

[79] Birman K, Cooper R, Joseph T, et al. The ISIS System Manual[R]. *Ithaca*: *Technical Report of Cornell University*.

[80] Birman K, Cooper R. The ISIS Project: Real Experience with a Fault Tolerant Programming System[J]. *ACM SIGOPS Operating Systems Review*, 1991, 25(2): 103-107.

[81] Verboncoeur J P, Alves M V, Vahedi V, et al. Simultaneous Potential and Circuit Solution for 1D Bounded Plasma Particle Simulation Codes[J]. *Journal of Computational Physics*, 1993, 104(2): 321-328.

[82] Forslund D W, Wingate C, Ford P, et al. Experiences in Writing a Distributed Particle Simulation Code in C++[C]. *San Francisco*: 1990 *USENIX C++ Conference*, 1990.

[83] Forslund D W, Wingate C A, Ford P S, et al. A Distributed Particle Simulation Code in C++[C]. *Dallas*: *American Society of Civil Engineers*(*ASCE*) *Conference*, 1992.

[84] Yuan T, Ren J, Zhou J, et al. The Effects of Numerical Acceleration Techniques on PIC-MCC Simulations of Ion Thrusters[J]. *AIP Advances*, 2020, 10(4): 045115.

[85] Chen Z, Wang Y, Ren J, et al. The Fully-Kinetic Investigations on the Ion Acceleration Mechanisms in an Electron-Driven Magnetic Nozzle[J]. *Plasma Sources Science and Technology*, 2022, 31(5): 055013.

[86] Chen Z, Wang Y, Tang H, et al. Compositions and Distributions of the Azimuthal Currents in the Magnetic Nozzle[J]. *Plasma Sources Science and Technology*, 2021, 30(10): 105012.

[87] Chen Z, Wang Y, Tang H, et al. Electric Potential Barriers in the Magnetic Nozzle[J]. *Physical Review E*, 2020, 101(5): 053208.

[88] Jiang Y, Tang H, Ren J, et al. Magnetic Mirror Effect in a Cylindrical Hall Thruster[J]. *Journal of Physics D*: *Applied Physics*, 2018, 51(3): 035201.

[89] Cao S, Ren J, Tang H, et al. Modeling on Plasma Energy Balance and Transfer in a Hollow Cathode[J]. *Journal of Physics D*: *Applied Physics*, 2019, 52(28): 285202.

[90] Cao S, Ren J, Tang H, et al. Numerical Simulation of Plasma Power Deposition on Hollow Cathode Walls Using Particle-in-Cell and Monte Carlo Collision Method[J]. *Physics of Plasmas*, 2018, 25(10): 103512.

[91] 曹 帅. 电推力器钡钨空心阴极工作过程的数值模拟研究[D]. 北京: 北京航空航天大学, 2019.

[92] Decyk V K, Norton C D. UCLA Parallel PIC Framework [J]. *Computer Physics Communications*, 2004, 164(1): 80-85.

[93] Flynn M J. Some Computer Organizations and Their Effectiveness[J]. *IEEE Transactions on Computers*, 1972, 100(9): 948-960.

[94] Snir M, Otto S, Huss-Lederman S. MPI-the Complete Reference: The MPI Core[M]. *Cambridge*: *MIT Press*, 1998.

[95] Gropp W, Lusk E. User's Guide for MPICH: A Portable Implementation of MPI[M]. *Chicago*: *Argonne National Laboratory*, *Chicago University*, 1996.

[96] Gabriel E, Fagg G E, Bosilca G, et al. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation[C]. *Berlin*: *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, 2004.

[97] Mardahl P J. PIC Code Charge Conservation, Numerical Heating, and Parallelization: Application of XOOPIC to Laser Amplification via Raman Backscatter[D]. *Berkeley*: *University of California*, 2001.

[98] Mo Z, Zhang A, Cao X, et al. JASMIN: A Parallel Software Infrastructure for Scientific Computing[J]. *Frontiers of Computer Science in China*, 2010, 4(4): 480-488.

[99] 刘 旭. JASMIN框架面向亿亿次计算的研究进展[J]. 2016年版中国工程物理研究院科技年报, 2016(1): 54-59.

[100] 莫则尧. 高性能数值模拟编程框架研究进展[J]. 科研信息化技术与应用, 2015, 6(4): 11-19.

[101] Munshi A, Gaster B, Mattson T G, et al. OpenCL Programming Guide [M]. *Boston*: *Pearson Education*, 2011.

[102] Buck I. GPU Computing with NVIDIA CUDA[C]. *San Diego*: *ACM SIGGRAPH* 2007 *Courses*, 2007.

[103] Kirk D. NVIDIA CUDA Software and GPU Parallel Computing Architecture [C]. *Montreal*: 6*th International Symposium on Memory Management*, 2007.

[104] Stantchev G, Dorland W, Gumerov N. Fast Parallel Particle-to-Grid Interpolation for Plasma PIC Simulations on the GPU[J]. *Journal of Parallel and Distributed Computing*, 2008, 68(10): 1339-1349.

[105] 刘腾宇. 离子推进器粒子模拟的GPU加速研究[D]. 成都: 电子科技大学, 2018.

[106] Burau H, Widera R, Hönig W, et al. PIConGPU: A Fully Relativistic Particle-in-Cell Code for a GPU Cluster [J]. *IEEE Transactions on Plasma Science*, 2010, 38(10): 2831-2839.

[107] Decyk V K, Singh T V. Particle-in-Cell Algorithms for Emerging Computer Architectures[J]. *Computer Physics Communications*, 2014, 185(3): 708-719.

[108] 金晓林, 李 斌, 杨中海, 等. 快速粒子模拟软件BUMBLEBEE研制及其应用研究[C]. 厦门: 真空电子学分会第二十届学术年会, 2016.

[109] 何英杰. 1D3V粒子模拟软件BUMBLEBEE的多核TBB并行研究[D]. 成都: 电子科技大学, 2016.

[110] Ryabinkin A N, Serov A O, Pal A F, et al. Structure of DC Magnetron Sputtering Discharge at Various Gas Pressures: A Two-Dimensional Particle-in-Cell Monte Carlo Collision Study[J]. *Plasma Sources Science and Technology*, 2021, 30(5): 055009.

[111] Mankofsky A, Seftor J L, Chang C L, et al. Domain Decomposition and Particle Pushing for Multiprocessing Computers [J]. *Computer Physics Communications*, 1988, 48(1): 155-165.

[112] Di Martino B, Briguglio S, Vlad G, et al. Parallel PIC Plasma Simulation Through Particle Decomposition Techniques[J]. *Parallel Computing*, 2001, 27(3): 295-314.

[113] 姜 巍. 低温等离子体中的PIC/MC方法简介: 演化和稳态模拟[R]. 武汉: 华中科技大学, 2022.

[114] Zhang Z Q, Liu Q H. Applications of the BCGS-FFT Method to 3-D Induction Well Logging Problems [J]. *IEEE Transactions on Geoscience and Remote Sensing*, 2003, 41(5): 998-1004.

[115] Wang C F, Jin J M. Simple and Efficient Computation of Electromagnetic Fields in Arbitrarily Shaped Inhomogeneous Dielectric Bodies Using Transpose-Free QMR and FFT [J]. *IEEE Transactions on Microwave Theory and Techniques*, 1998, 46(5): 553-558.

[116] 廖 臣, 祝大军, 刘盛纲. 五点差分格式求解泊松方程并行算法的研究[J]. 电子科技大学学报, 2008, 37(1): 81-83.

[117] Van der Pas R. The PVM Implementation of a Generalized Red Black Algorithm [J]. *Supercomputer*, 1993, 10: 72-72.

[118] Li H L, Yuan X, Yang J, et al. Numerical and Theoretical Modeling of the Sheath Upstream of Ion Optics: Sheath Structure Transition and Its Effect on the Beam Divergence [J]. *Plasma Sources Science and Technology*, 2021, 30(7): 075019.

[119] Dudson B D, Umansky M V, Xu X Q, et al. BOUT++: A Framework for Parallel Plasma Fluid Simulations [J]. *Computer Physics Communications*, 2009, 180(9): 1467-1480.

[120] Ortwein P, Binder T, Copplestone S, et al. Parallel Performance of a Discontinuous Galerkin Spectral Element Method Based PIC-DSMC Solver [C]. *Stuttgart: High Performance Computing in Science and Engineering'14*, 2014.

[121] Ferraro R D, Liewer P C, Decyk V K. Dynamic Load Balancing for a 2D Concurrent Plasma PIC Code [J]. *Journal of Computational Physics*, 1993, 109(2): 329-341.

[122] Derouillat J, Beck A, Pérez F, et al. Smilei : A Collaborative, Open-Source, Multi-Purpose Particle-in-Cell Code for Plasma Simulation [J]. *Computer Physics Communications*, 2018, 222: 351-373.

[123] Othmer C, Schüle J. Dynamic Load Balancing of Plasma Particle-in-Cell Simulations: The Taskfarm Alternative [J]. *Computer Physics Communications*, 2002, 147(1-2): 741-744.

[124] Nakashima H, Miyake Y, Usui H, et al. Ohhelp: A Scalable Domain-Decomposing Dynamic Load Balancing for Particle-in-Cell Simulations [C]. *Yorktown Heights: 23rd International Conference on Supercomputing*, 2009.

[125] Assous F, Pougeard Dulimbert T, Segré J. A New Method for Coalescing Particles in PIC Codes [J]. *Journal of Computational Physics*, 2003, 187(2): 550-571.

[126] Teunissen J, Ebert U. Controlling the Weights of Simulation Particles: Adaptive Particle Management Using K-D Trees [J]. *Journal of Computational Physics*, 2014, 259: 318-330.

[127] Korkut B, Li Z, Levin D A. 3-D Simulation of Ion Thruster Plumes Using Octree Adaptive Mesh Refinement [J]. *IEEE Transactions on Plasma Science*, 2015, 43(5): 1706-1721.

[128] Ricker P M. A Direct Multigrid Poisson Solver for Oct-Tree Adaptive Meshes [J]. *The Astrophysical Journal Supplement Series*, 2008, 176(1).

(编辑:梅 瑛)