格子Boltzmann方法多层网格负载均衡算法优化研究

何 鹏[1],王良军[1],张 武[2],朱文浩[1*]
(1. 上海大学计算机工程与科学学院,上海 200444;2. 上海大学力学与工程科学学院,上海 200072)

摘 要: 基于格子Boltzmann方法的多层网格局部加密技术,通过多尺度网格计算不同层次的流动特征,避免了单层均匀笛卡尔网格中的低效率与计算资源的浪费,但仍存在并行性能上的不良影响. 本文考虑并行计算中的负载均衡效应,从单层网格出发,通过考虑多层网格的运算特点来研究基于负载均衡的网格划分方法. 同时,将网格划分与程序实现进行分离,在单层和多层网格中均完成了任意网格划分下的并行计算. 在单层网格中,以二维血管流动的不同并行策略为例,研究了负载量划分与不同进程的各自时间开销的关系. 在多层网格中,首先论述了多尺度网格在运算顺序上的特征,其次以三种不同的多层网格验证二维翼型绕流的计算结果,最后在每种网格中均使用三种不同的网格划分方法进一步探讨负载均衡与时间开销的关系. 在128核的高性能计算平台上进行并行性能测试,强可扩展性可达到60%左右,弱可扩展性可达到82.78%. 这种高可扩展性结果表明本文通过改进负载均衡性能,明显提升了多层网格计算中的并行性能.

关键词: 格子Boltzmann方法;多层网格;大规模并行计算;负载均衡
基金项目: 国家自然科学基金(No.61873156,No.91630206)
中图分类号: TP312 **文献标识码:** A **文章编号:** 0372-2112(2024)09-3097-14
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20230809

Optimization Study of the Load Balancing Algorithm in the Multi-Layer Lattice Boltzmann Method

HE Peng[1], WANG Liang-jun[1], ZHANG Wu[2], ZHU Wen-hao[1*]
(1. *School of Computer Engineering and Science*, *Shanghai University*, *Shanghai* 200444, *China*;
2. *School of Mechanics and Engineering Science*, *Shanghai University*, *Shanghai* 200072, *China*)

Abstract: The local encryption technique for multi-layer grids based on the lattice Boltzmann method computes the flow characteristics at different levels through multi-layer grids, which avoids the inefficiency and waste of computational resources in single-layer uniform Cartesian grids. But there is still an undesirable effect on the parallel performance. The load-balancing effect in parallel computing is considered in this paper. Starting from a single-layer grid, we study the load-balancing-based grid partitioning method by considering the computational characteristics of multi-layer grids. At the same time, the grid partitioning is separated from the program implementation, and parallel computation with arbitrary grid partitioning is achieved in both single-layer and multi-layer grids. The relationship between load partitioning and the respective time overheads of the different processes is investigated in a single-layer grid with different parallel strategies for 2D vascular flow. The characteristics of multiscale grids with respect to the order of operations is first discussed for multi-layer grids. Second, three different multi-layer grids are used to verify the computational results of the two-dimensional aerofoils. Finally, the relationship between load balancing and time overhead is further investigated by using three different meshing methods in each grid. Parallel performance tests on a 128-core HPC (High Performance Computing) platform show that the strong scalability can reach up to 60%, and the weak scalability can reach 82.78%. This high scalability result shows the significant improvement of the parallel performance in multi-layer grid computing by improving the load balancing performance.

Key words: lattice Boltzmann method; multilayer grids; massive parallel computing; load balancing

收稿日期:2023-08-26;修回日期:2024-02-01;责任编辑:梅志强
*通讯作者:朱文浩

Foundation Item(s): National Natural Science Foundation of China (No.61873156, No.91630206)

1 简介

复杂流体运动问题一直是大规模科学与工程计算中最重要且具有挑战性的研究领域之一. 从Boltzmann方程出发得到的格子Boltzmann方法(Lattice Boltzmann Method,LBM),以介观尺度描述物理状态中的流体系统和热力学现象,是计算流体力学(Computational Fluid Dynamics,CFD)中一个成熟的数值模拟方法[1,2]. 在LBM的模型发展中,Chen和Qian等人分别独立提出了单松弛时间(Single Relaxation Time,SRT)或BGK(Bhatnagar-Gross-Krook)模型[3,4]. 由于该模型使用单一松弛系数的简单性和有效性,BGK模型是最广泛接受的格子Boltzmann模型之一. 在单松弛时间格子Boltzmann模型(Single Relaxation Time-Lattice Boltzmann Method,SRT-LBM)的计算中,借助一些湍流模型方法,可以克服计算稳定性较差的缺点. 基于涡粘模型理论,Hou等[5]将Smagorinsky模型引入SRT-LBM中,有效地提高了SRT-LBM的雷诺数限制,提高了算法的灵活性. 其次,由于笛卡尔网格结构的简单性,基于笛卡尔网格的LBM可以很容易地进行计算规模上的扩展,对于实际问题中,通过网格分割和修补后的复杂几何模型[6],可以进行灵活的边界条件处理[7,8]. 这些特性使LBM被认为是模拟复杂流体系统的一种有效方法.

网格是数值模拟的基础,由于笛卡尔网格的正交性可简化算法中的部分公式,可以降低计算的复杂性并较准确的捕捉流动结构特征[9]. 然而均匀的网格不太适合模拟存在强烈局部梯度的流体流动. 为了将计算资源集中在需要高精度网格分辨率的区域,CFD的许多先进方法都依赖于局部网格细化. Filippova和Hänel首先将局部网格细化处理引入LBM[10],其他学者在后续的研究中对这一概念进行完善和发展[11,12]. 多尺度网格可以更好地获得不同区域的细节信息[13]. 不同尺度的网格在交界处存在数据交换,因而该方法在粗、细网格上的演化是双向耦合的,可以同时平衡计算效率和计算精度. 对于基于格心的网格细化,Neumann等人[14]和Hasert等人[15]提出了适合大规模集群系统的并行化方法. 在分区方法中将空间填充曲线应用于并行框架中,使得并行系统具有良好的扩展性. 相比之下,Schornbaum等人在他们的研究工作中要求将各级网格单元分布到所有提供的平行域中,以实现计算任务的负载均衡[16]. Liu等人[17]设计了一种新的网格划分来代替图分区库Metis,以实现进一步的负载均衡. 本文的方法基于格点格式的网格细化,将分区方法与并行计算独立开来,以发展基于负载均衡的分区策略与基于任意分区状态下的并行方法.

对本文提出的新的并行方法进行了实验验证. 首先验证了单层网格中网格的进程分区与时间开销的关系,负载均衡的单层网格下的流场计算具有最快的运算性能. 基于格点格式的多层局部网格细化的LBM被称为多层网格LBM. 在这样的网格结构上,本文探索了该结构的计算特性,并给出一种基于负载平衡的并行算法来提高多层网格LBM的计算效率. 本文进行了一系列的数值模拟实验,并在不同的高性能计算平台上验证了本文中的多层网格LBM及其并行方案的准确性和稳定性. 此外,由于本文将网格划分方法与并行计算相互独立,为今后发展复杂的多层网格细化方法和与之相关的负载均衡技术打下了应用基础.

2 数值方法与网格结构

2.1 LBM中的控制方程

与传统的模拟方法不同,LBM通过计算大量粒子在空间中的单独运动,从而确定物理系统中的主要宏观物理量状态. 基本思想并不是为了确定单独粒子的运动状态,而是要找到每个分子处于某种状态的概率,通过统计来得出系统的宏观物理量[18]. 一个基本的LBM算法框架如算法1所示.

算法1 LBM的计算框架

输入:初始流场状态(密度、速度和雷诺数等等);
输出:最终的流场计算结果;
1:初始化;
2:网格生成;
3:碰撞;
4:迁移;
5:计算计算域内的固体边界条件;
6:计算宏观量;
7:计算整体计算域的外边界条件
8:判断是否达到计算条件,如果没有,则返还到步骤3;
9:输出结果并保存文件.

在单松弛时间模型中,碰撞和迁移过程是对离散后的Boltzmann方程进行求解.

$$f_i(\boldsymbol{x}+c\boldsymbol{e}_i\Delta t,t+\Delta t)=f_i(\boldsymbol{x},t)-\frac{1}{\tau}[f_i(\boldsymbol{x},t)-f_i^{\mathrm{eq}}(\boldsymbol{x},t)] \quad (1)$$

其中,τ是碰撞的特征松弛时间,常数$c=\Delta x/\Delta t$,Δx和Δt分别为格子尺寸和时间步长,不同方向上的格子尺寸一致. i是离散速度方向,$\boldsymbol{e}_i$是离散速度,f_i^{eq}和f_i分别是i方向上的平衡态分布函数与分布函数. 不同的模型中采用的$\boldsymbol{e}_i$与f_i^{eq}存在差异. 在D2Q9模型中,

$$\boldsymbol{e}_i=\begin{cases}(0\quad 0), i=0\\ c\left(\cos\left[(i-1)\dfrac{\pi}{2}\right]\quad \sin\left[(i-1)\dfrac{\pi}{2}\right]\right)\\ \qquad\qquad , i=1,2,3,4\\ \sqrt{2}\,c\left(\cos\left[(2i-1)\dfrac{\pi}{4}\right]\quad \sin\left[(2i-1)\dfrac{\pi}{4}\right]\right)\\ \qquad\qquad , i=5,6,7,8\end{cases} \tag{2}$$

$$f_i^{\mathrm{eq}}=\rho\omega_i\left[1+\frac{\boldsymbol{e}_i\cdot\boldsymbol{u}}{c_s^2}+\frac{(\boldsymbol{e}_i\cdot\boldsymbol{u})^2}{2c_s^4}-\frac{\boldsymbol{u}^2}{2c_s^2}\right] \tag{3}$$

其中,$\boldsymbol{u}$为宏观量速度,ρ为密度,c_s为格子声速,ω_i为权系数. 对于宏观量的定义如下:

$$\begin{cases}\rho=\sum_i f_i\\ \boldsymbol{u}=\dfrac{1}{\rho}\sum_i f_i\boldsymbol{e}_i\\ p=\rho c_s^2\end{cases} \tag{4}$$

2.2　流场边界条件处理

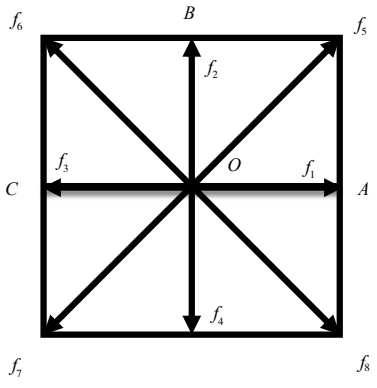
在数学上,通常使用边界条件来描述周围环境对流体流动的影响. 在计算域内部,经过碰撞和迁移过程后得到网格单元上的分布函数,进而在宏观量计算的过程中得到宏观量状态. 然而在边界附近的部分分布函数是未知的,因此无法得到宏观量状态. 边界处理对数值计算的准确性、稳定性和效率存在很大影响. 在本文的论述中,根据使用边界条件的区域不同,分为外边界条件和内边界条件. 对于外边界条件,通常指整体计算域和计算域之外的边界,例如在二维中是指包含几何体在内的矩形计算域的四条线段,而三维中指立体计算域的六个面. 外边界条件通常采用非平衡态外推格式[19],数学形式如下:

$$f_i(O,t)=f_i^{\mathrm{eq}}(O,t)+[f_i(B,t)-f_i^{\mathrm{eq}}(B,t)] \tag{5}$$

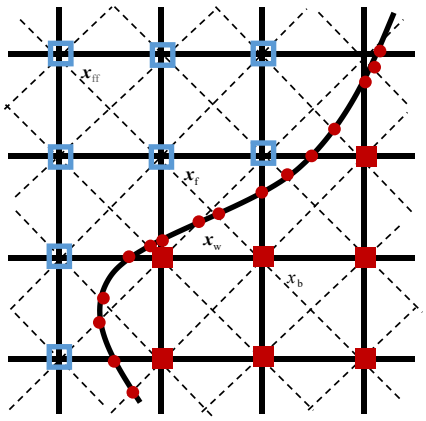
其中,点O和点B分别为边界节点和边界邻居点,如图1(a)所示. 如果节点O上存在未知宏观物理量,则由B点的相应值代替,以速度边界条件为例,已知O点速度,密度未知,则O点密度取B点密度. 对于内边界条件,根据处理实际问题的边界条件类型,可分为直线边界和曲线边界. 而曲线边界又可包含直线边界,因此仅介绍本文使用的曲线边界条件. 对于一些绕流问题,在计算域的内部存在固体几何体,固体包含的网格点不参与流场计算. 在邻近固体的网格点在迁移过程中存在未知分布函数,因此需要内边界条件. 内边界条件采用统一边界处理格式[20],数学形式如下:

$$\begin{aligned}f_{\bar{i}}(\boldsymbol{x}_{\mathrm{f}})=\frac{1}{1+q}[&qf_{\bar{i}}(\boldsymbol{x}_{\mathrm{f}})+qf_i(\boldsymbol{x}_{\mathrm{f}})\\ &+(1-q)f_i(\boldsymbol{x}_{\mathrm{ff}})+2\omega_i\rho_{\mathrm{f}}\frac{3}{c^2}\boldsymbol{e}_i\boldsymbol{u}_{\mathrm{w}}]\end{aligned} \tag{6}$$

其中$\boldsymbol{u}_{\mathrm{w}}$表示曲线边界的运动速度,方向$\bar{i}$为方向$i$的反方向,相对距离$q=|\boldsymbol{x}_{\mathrm{f}}-\boldsymbol{x}_{\mathrm{w}}|/|\boldsymbol{x}_{\mathrm{f}}-\boldsymbol{x}_{\mathrm{b}}|$,点$\boldsymbol{x}_{\mathrm{f}}$、点$\boldsymbol{x}_{\mathrm{ff}}$、点$\boldsymbol{x}_{\mathrm{w}}$和点$\boldsymbol{x}_{\mathrm{b}}$分别为邻近固体边界的流体点、邻近前者的流体点、边界上的固体点和邻近固体边界的固体点,节点类型如图1(b)所示.



(a) 非平衡态外推格式的边界条件,阴影表示固体壁



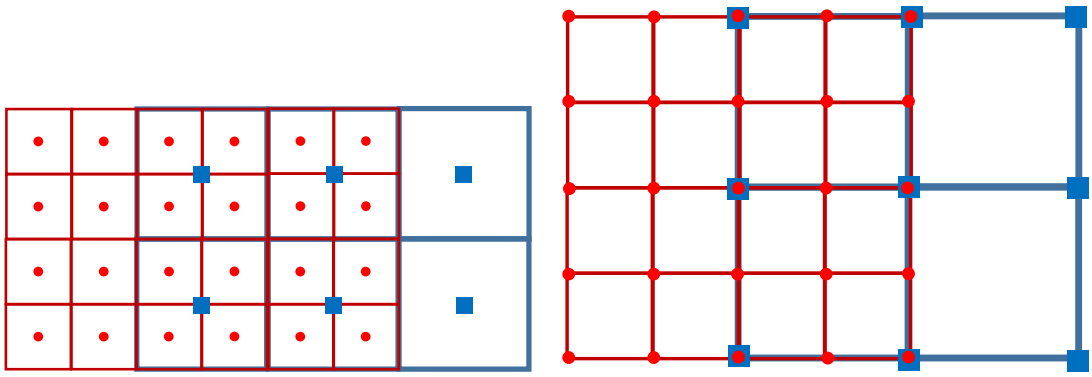
(b) 复杂边界上的节点类型,黑色曲线表示固体壁

图1　二维的问题中的平直边界处理与曲边界处理

2.3　基于格点格式的多层网格

由于LBM基于对称的离散速度模型,而笛卡尔网格存在对称性和正交性,因此最适合采用笛卡尔网格作为计算网格. 然而当计算域内存在物理量急剧变化的区域时,整个计算域中使用均匀网格计算时通常需要较大的网格量. 为了解决这个问题,在CFD中经常使用局部网格加密的方式,通过更细的网格捕捉到急剧变化区域的流动信息. 根据物理量在网格上的存储位置不同,多层网格存在两种形式:格心格式(图2(a))和格点格式(图2(b)). 对于单个网格,当物理量存储在格子中心时,称为格心格式,当物理量存储在格子点上时,称为格点格式. 两种格式

在单层网格中的区别较小,在多层网格中区别较大,主要体现在程序实现与不同尺度网格的数据交换处.在图2(*a*)中,虽然格心格式可以利用到一个粗格子对应四个细格子的四叉树结构,在程序实现中更加方便.然而在粗细格子传递数据的时候,不同于格点法的粗格子与细格子的点对点直接传输,而是需要先插值到相同的空间位置处.基于上述分析,本文采用格点格式来构建多层网格系统.



(*a*) 格心格式 (*b*) 格心格式

图2 不同格式的多层网格结构

本文提出一种简便的构建多层网格方式,通过不同尺寸的网格依次堆叠形成多层网格,如图3所示.由于构建单层的整体网格是非常容易的,根据每层网格的网格量和网格尺寸不同,依次堆叠形成图3中的三层网格.

图3中最粗网格尺寸的称为0层网格,如黑色网格所示,随着网格尺寸逐渐减小,网格层数标号递增.在数据交换的过程中,最初是采用时间插值得到细格子的半个时间步上的数据,同一时刻上相同位置的粗细格子通过对应公式进行交换[10].后面的研究多采用额外设置“缓冲区”[21]的方法,因为这种方法可以根据LBM的计算特点来避开细格子的时间插值步骤.在算法1的LBM计算框架基础上,给出多层网格LBM的计算框架,如算法2所示.与串行计算不同,在MPI并行计算中,“缓冲区”的存在虽然避免了时间插值,但会导致在一个时间步的过程中,不同进程之间需要进行多次通信.如果直接对算法2进行MPI并行化,并行程序的并行性能通常较低.

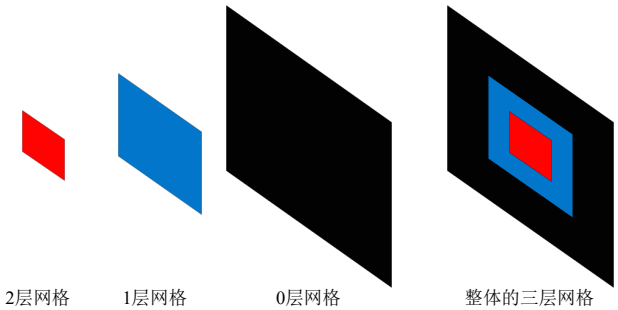


图3 多层网格的构建方法

算法2 任意层数下的多层网格LBM的计算框架

输入:初始流场状态(密度、速度和雷诺数等)与计算网格(网格层数N,每层网格用k表示);

输出:最终的流场计算结果;

1:初始化,每层网格的常数mark设为0;

2:$n=1$;

3:$m=1$;

4:对第$N-1$层网格进行LBM运算(包含碰撞、迁移、计算内边界和宏观量计算),同时对应的mark加1;

5:对第$N-1$层网格进行LBM运算,对应的mark加1;

6:$l=N-1$;

7:判断第l层的mark是否等于2,如果否,则直接跳至步骤10;

8:对第l层网格进行LBM运算,对应的mark加1;

9:交换第l层和第$l-1$层的缓冲区数据(包含宏观量和分布函数),并将第l层的mark设为0;

10:$l=l-1$,判断l是否小于1,如果否,则返回到步骤6;

11:$m=m+1$,判断m是否大于2^{N-2},如果否,则返回到步骤4;

12:计算第0层网格中的外边界条件;

13:$n=n+1$,判断n是否达到计算步数,如果否,则返回到步骤3;

14:输出计算结果.

3 并行计算中的网格划分与数据通信

由于标准LBM只需要和邻居点交换数据,即只进行局部通信,通常具有良好的可扩展性.但对于计算域内存在大量固体区域的流体流动,还需要考虑进程间的负载均衡问题.对于单层网格的流场计算,负载平衡的任务划分是实现较高并行效率的基础.对于局部网格细化的多层网格,还需要对负载均衡进行进一步的优化.在本节中,提出了一种常规矩形计算域中的网格划分方式与自定义划分,同时给出对于任意网格划分

情况下的并行计算策略,以实现更短的运算时间.

3.1　负载的常规划分与自定义划分

对于通常的矩形计算域,以三维网格为例,如图4中沿着不同的方向对网格进行切割,形成一维均匀划分法、二维均匀划分法和三维均匀划分法. 根据LBM的计算特点,邻近相邻子网格的区域数据以MPI(Multi Point Interface)通信协议进行数据通信. 以数学公式的形式进行描述,若网格在三个维度上的数量分为n_x,n_y和n_z,将整体任务分为n_p份进程,则每个进程的子网格总量为$\left(n_x/n_p\right)\times n_y\times n_z$. 依次类推,二维均匀划分法和三维均匀划分法就是在其余方向上的网格数量进行同样的均匀划分. 负载划分的维度越高,那么可划分的进程数就越多. 相应的,从图4中红色区域即通信区的状态可以看出,划分方式越复杂,构建子网格间的通信区则越困难.

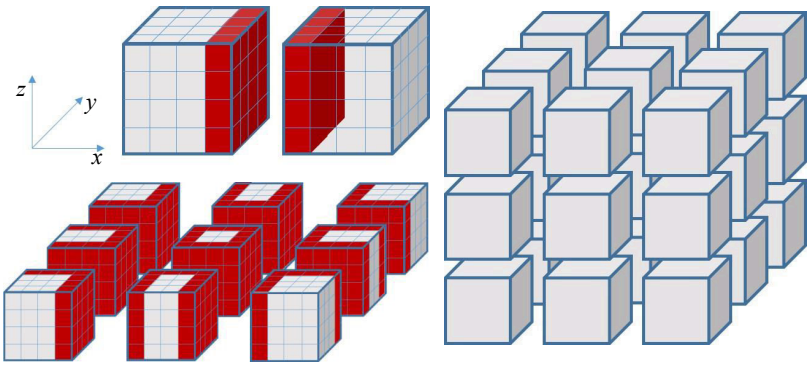


图4　常规的1D,2D和3D网格划分方法

常规划分适合于矩形计算域的并行计算,例如一维均匀划分法,通信区在子网格上靠近邻局子网格的区域. 因此,只需要在子网格外部额外扩充一列网格,用于接收邻居子网格数据即可. 对于自定义网格划分方法,则是研究者根据需要,按照一定的规则对网格进行切割以实现更快的并行计算. 考虑负载均衡的特定网格划分方式,通常是为了获取更加优良的负载均衡效果,然而由于"通信区"的构建方法不同,还应考虑程序实现的影响. 对于单层网格,基于负载均衡效果的网格划分方式如算法3所示.

对于多层网格的网格划分,与单层网格类似,不同的是每一层均是独立进行网格划分,同时先划分计算区域再划分缓冲区域. 本文为了对负载均衡效果进行评估,将同步率定义为:

$$L=1-\left|\frac{T_{\mathrm{loc}}-T_{\mathrm{avg}}}{T_{\max}}\right| \tag{7}$$

通过分析全部进程的同步率差异,可以更加清晰地显示整体任务运算的均衡性. 其中,L为同步率,T_{loc}为一个进程的运算时间,T_{avg}为全部进程的平均运算时间,$T_{\max}$为全部进程的最大运算时间.

算法3　基于负载均衡的单层网格中的网格划分方法

输入:计算网格(已完成网格生成步骤,即已确定网格点类型和相对距离);

输出:考虑负载均衡的网格划分结果(在原计算网格上新增一个标记常数,表示该点在并行计算中的进程编号);

1:对网格数为$n_x\times n_y$的二维网格,创建一维整型数组nn(n_x),存储统计的每一列的计算点数量;

2:如果整体任务划分给n_p个进程,则累加全部计算点数量并除以n_p,得到平均数avg. 创建一维整型数组mark,数组长度从0到n_p-1,存储进程分割线所在的列编号;

3:对于常数m和0号进程,令$m=0$,从第1列开始累加每一列的计算点数量. 如果计算点数量小于等于$(m+1)\cdot$avg,同时大于等于$(m+1)\cdot$avg减去下一列的计算点数量,则令mark(m)等于当前列编号;

4:重复步骤3,直到已得到mark(n_p-2),令mark(n_p-1)$=n_x$;

5:根据mark即可确定全部网格点的进程编号;

6:输出结果并保存文件.

3.2　任意网格划分下的多层网格LBM并行算法

由于网格结构和粗细网格数据传输方式的不同,频繁的粗细网格数据传输意味着进程间频繁的数据交换,当进行大规模计算时,如不考虑负载均衡将会显著降低并行性能. 本文通过独立实施网格划分和并行方法,对于任意网格划分情况下的网格均可进行计算,使得可以方便地实现更高并行性能的网格划分方法. 对于经过网格划分后的任意网格,都是在原网格上加入一个标记数组,该数组保存了所有网格点对应的进程编号. 从这个结果出发,到多层网格LBM并行计算之前,还需要一步预处理步骤,用于构建不同进程之间的数据传输通道.

预处理步骤如下:读取计算网格后,对于邻居点在其余进程的网格点,根据网格划分情况额外扩增缓冲区,用于存储接收的其余进程的对应点的数据. 将扩充的缓冲区变换为一维形式增加在计算网格后,再根据缓冲区情况得到对应的数据发送点的坐标信息,即可构建数据传输通道.

在并行计算中,当需要进行不同进程之间的数据通信时,根据数据传输通道即可确定需要传输的网格点坐标和接收数据的网格点坐标.确定坐标后对需要传输的数组进行打包,传输完毕后再进行解包,即可完成不同进程之间的数据通信.对于任意网格划分下的多层网格LBM并行算法如算法4所示.

算法4 任意网格划分下的多层网格LBM并行算法

输入:初始流场状态(密度、速度和雷诺数等等)、计算网格(网格层数N,每层网格用k表示)和数据传输通道(接收和发送的网格点数量以及坐标);

输出:最终的流场计算结果;

1:根据MPI进程总数n_p和读入的文件,从第$N-1$层网格到第0层创建每个进程需要的网格点数,同时保存每个网格点的对应坐标;

2:根据每一层的网格点坐标,构建不同层网格点的重合关系,即构建粗细网格交换的"缓冲区";

3:每个MPI进程负责自己的网格区域,参考算法2进行LBM运算;

4:每完成相邻两层的运算后,先基于数据传输通道完成不同进程之间的数据通信过程;

5:进行粗细网格的数据交换;

6:计算第0层网格的外边界条件;

7:判断是否达到计算条件,如果没有,则返还到步骤3;

8:输出结果并保存文件.

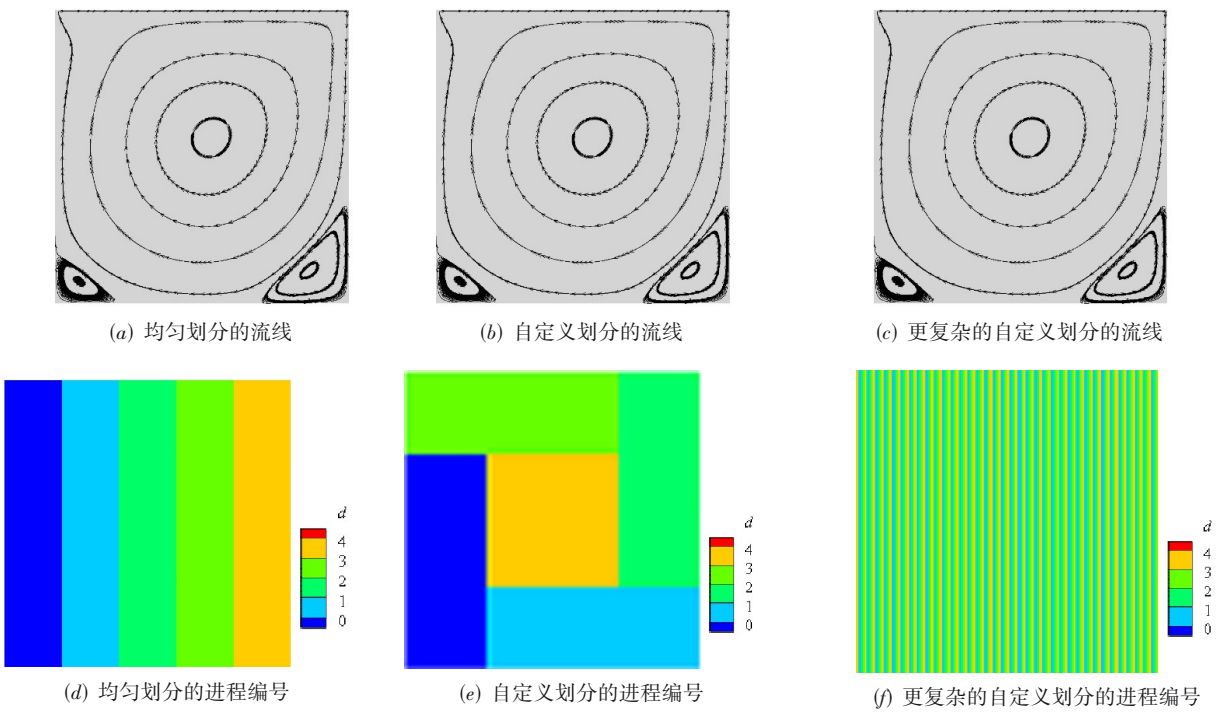
4 单层网格中的网格划分

本文在常规的网格划分方法的基础上,总结了MPI并行计算中数据通信的特点,实现了任意网格划分下的并行计算.在单层网格的计算中,通过不同网格划分方式下的顶盖驱动流,分析任意网格划分下的通信量影响.通过不同网格划分方式下的二维血管流,在验证该方法的同时,讨论了单层网格中网格划分与负载均衡的关系.

4.1 不同网格划分下的数据通信量影响

在顶盖驱动流问题中,由于流体计算域为矩形结构,网格划分时的均匀划分法通常会有较好的负载均衡效果.在矩形计算域的流体问题中,通过不同进程划分的网格数量表示不同进程划分的计算负载.本节通过顶盖驱动流的计算,讨论不同网格划分下的数据通信量影响.由于可实现任意网格划分状态下的顶盖驱动流MPI并行计算,因此可以排除程序实现的不同造成的影响.在三种不同网格划分状态下的顶盖驱动流计算中,流场的参数设置相同,仅并行计算时的网格划分状态不同.计算域的网格数量为400×400,雷诺数Re为1 000,顶盖移动速度为0.1,计算15万步,收敛判据小于10^{-6},全物理量采用无量纲化形式.使用的计算平台为128核的单结点集群,CPU型号为AMD EPYC 7702 64-Core Processor.在本文实现的并行程序中,用户可自定义网格点的进程归属.将整体任务划分给5个进程,使用三种不同的网格点划分方式,如图5所示.

图5(*d*)中按照x方向进行一维均匀划分.图5(*e*)



(*a*) 均匀划分的流线 (*b*) 自定义划分的流线 (*c*) 更复杂的自定义划分的流线

(*d*) 均匀划分的进程编号 (*e*) 自定义划分的进程编号 (*f*) 更复杂的自定义划分的进程编号

图5 三种分区策略下的顶盖驱动流计算,黑色曲线表示流线,不同颜色代表不同的进程编号值

中进行人为指定的块划分. 图5(*f*)中每隔1列划分给1个进程,依次循环形成较为复杂的进程划分状态. 由于图5(*a*)到图5(*c*)的结果是一致的,因此任意的网格划分方式仅影响MPI中的数据通信过程与每个进程的负载量. 从表1中可以看到,均匀划分与自定义划分的负载基本一致,两者的时间开销也基本一致. 而均匀划分和复杂的自定义划分的负载基本一致,由于后者的数据通信量显著增加,因此在时间开销上略有提升. 任意的网格划分对计算结果无影响,仅影响整体任务的时间开销,而当进程数量较少时复杂的分区方式下带来的通信问题影响相对较小. 本文进行不同进程数进行了多次测试,相关参数与测试结果如表1所示.

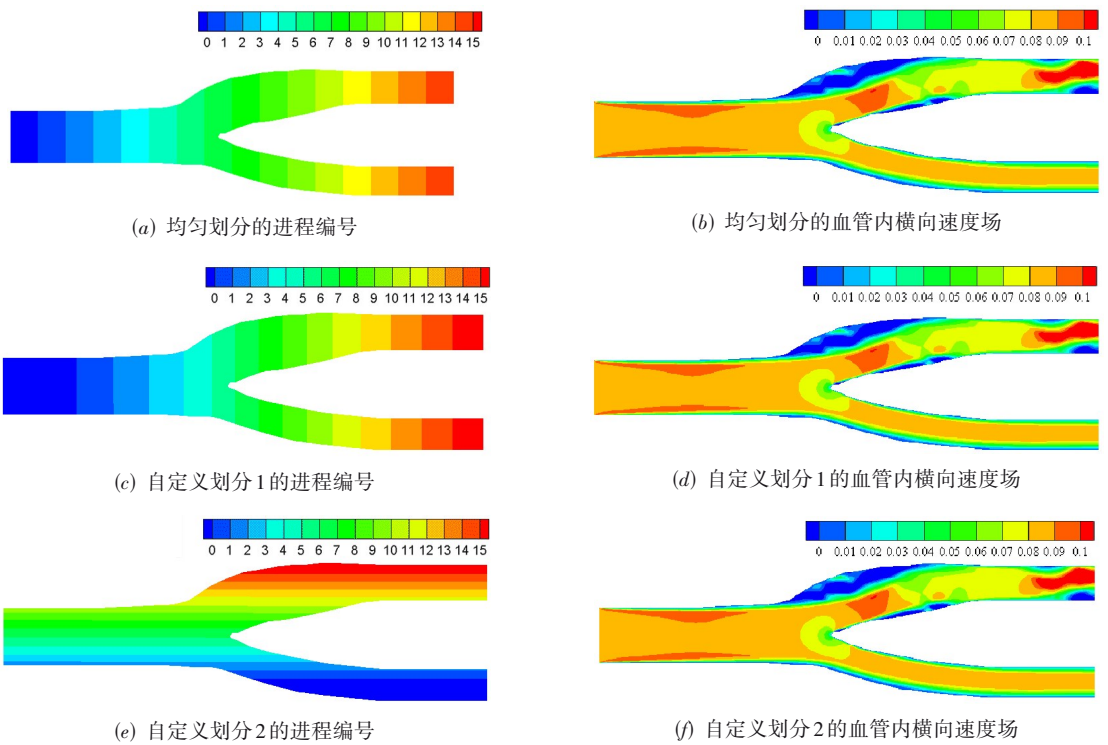
表1　不同分区策略下的相关参数与测试结果

	自定义划分	均匀划分	复杂的自定义划分
负载(最小/最大)(5进程)	31 684 /32 079	32 000 /32 000	32 000 /32 000
时间开销 (5进程)	25.41 s	25.61 s	29.84 s
最大通信量 (5进程)	716	800	64 000
负载(最小/最大)(50进程)	—	3 200 /3 200	3 200 /3 200
时间开销 (50进程)	—	3.13 s	3.52 s
最大通信量 (50进程)	—	800	6 400
负载(最小/最大)(100进程)	—	1 600 /1 600	1 600 /1 600
时间开销 (100进程)	—	2.24 s	2.39 s
最大通信量 (100进程)	—	800	3 200

4.2　单层网格中的负载均衡

在二维流体问题计算中,若矩形计算域中存在固体几何体,进行流体计算时需要通过网格生成找出对应的固体和流体网格点. 在计算过程中,仅对流体网格点进行计算,流体和固体边界处采用曲线插值. 因此对于这一类问题,例如本节中计算的血管流,通过不同进程划分的流体网格点数量表示每个进程划分的计算负载. 当矩形计算域中存在固体几何体时,常规的网格点划分方式容易造成不同进程划分的计算负载存在较大差距,以本节的二维血管流为例,如图6所示.

在图6的血管流问题中,血管几何模型为参照相关文献[22]描绘的二维分叉血管轮廓,用于讨论不规则几何的内流问题. 矩形计算域网格为2 500×1 000,流体网格量为89万左右,左端入口的来流速度为0.082 m/s,雷诺数为3 720,计算步数为30万步,血管壁处采用曲线插值,出口采用自由边界,计算平台与3.1节相同,整



(*a*) 均匀划分的进程编号　(*b*) 均匀划分的血管内横向速度场

(*c*) 自定义划分1的进程编号　(*d*) 自定义划分1的血管内横向速度场

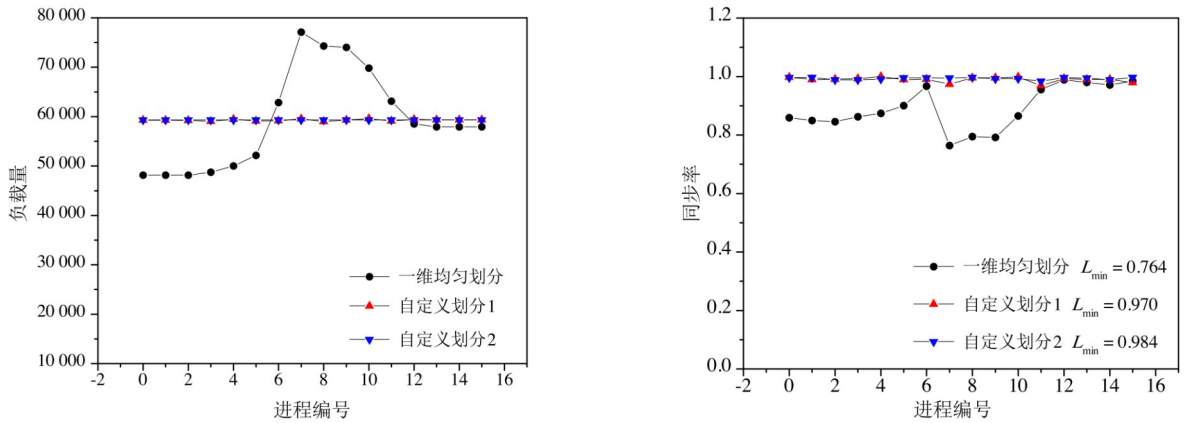
(*e*) 自定义划分2的进程编号　(*f*) 自定义划分2的血管内横向速度场

图6　不同网格点划分方式下的血管流计算结果

体任务划分给16个进程. 三种划分方式分别为一维均匀划分和两种自定义划分,其中自定义划分存在多种划分方式. 该节的两种自定义划分方式为,以一维均匀划分为基础,通过调整进程分界线的位置,从而降低使得不同进程内负载量的差异. 不同的是,自定义划分1的方式是以x轴方向的一维均匀划分为基础进行调整,自定义划分2的方式是以y轴方向的一维均匀划分为基础进行调整.

从图6中的计算结果可以看到,当管径变化较小时,如分叉后的下方血管,流动近似呈现为泊肃叶流动,当管径变化较大或者存在凹陷时,流动出现部分小涡,凹陷处存在低速回流状态. 通过流场分析可以验证血管流计算的正确性,同时对比均匀划分和两种自定义划分的流场结果,可以再次验证网格点划分方式不会影响到计算结果. 三种划分方式的计算时间分别为2 952 s、7 210 s、和7 560 s,不同的划分方式均使用相同程序进行计算,排除程序实现的干扰,说明网格划分会影响到整体的时间开销.

在图6中的三个计算中,均匀划分的时间开销大于两种自定义划分的时间开销,两种自定义划分的时间开销基本一致. 为了进一步说明这种差异是由于负载不均衡导致,在3.1节中提出同步率的定义,可以直观地表现出不同进程的时间开销. 理论上,当计算达到完美的负载均衡时,不同进程的同步率均等于1. 对比图7中的不同进程的负载量和同步率,发现当进程划分的负载量偏离平均负载量越大时,对应的进程同步率越低. 该结果说明在单层网格的并行计算过程中,计算网格划分是导致产生负载均衡的主要原因. 因此,本文可通过改变网格点的进程标识,改变不同进程划分的负载量,从而平衡不同进程的时间开销,降低整体的时间开销. 而任意网格划分下的程序设计可以实现自定义划分状态下的MPI并行计算.



(*a*) 负载量 (*b*) 同步率

图7 血管流中不同网格点划分状态下的负载量和同步率

5 多层网格中的网格划分

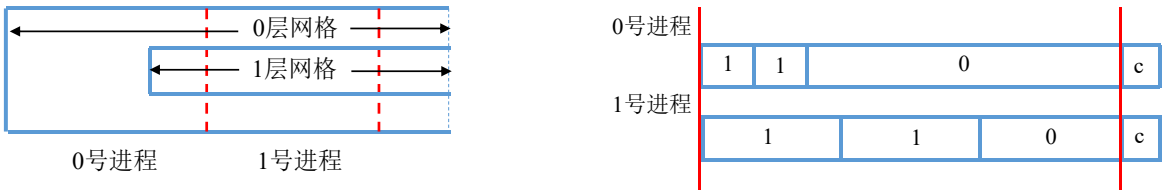
与单层网格相比,当计算需要更加精细的网格分辨率时,多层网格通过加密局部区域的网格,在完成计算需求的同时降低整体的网格需求. 然而在并行计算中,多层网格的并行性能通常较弱,制约了大规模的计算问题. 在本章中通过网格点的自定义划分,研究多层网格中的负载均衡问题和扩展性等并行计算性能问题.

5.1 不同尺度网格间的计算时序性

在多层网格的计算过程中,每一层网格内的计算相互独立,不同层之间通过"缓冲区"交换数据. 这种方法虽然避免了时间插值的过程,但带来了一个时序性的问题. 本文的时序性定义为,在三层及以上层数的多层网格系统中,不同区域的网格在计算时存在先后顺序,该顺序由高层网格区域指向底层网格区域. 由于串行计算是顺序计算,这个时序性在串行计算中无影响. 但在并行计算中,当不同进程划分的网格点分别位于高层网格区域和底层网格区域,时序性会导致不同进程出现等待过程.

在第4节中得到一个结论:单层网格中不同进程的时间开销主要受各自进程的负载量影响. 在多层网格中,在一维均匀划分的基础上,通过调整进程分割线可降低不同进程之间的负载量差异. 其中,通过网格点数量与权重的乘积表示计算负载,对于第n层的网格点,其权重为2^n. 在2.3节中介绍的多层网格系统由多层不同尺寸的网格堆叠而成,因此也称为"多层网格". 对于图8(a)的两层网格,红色虚线是进程分割线,将整体网

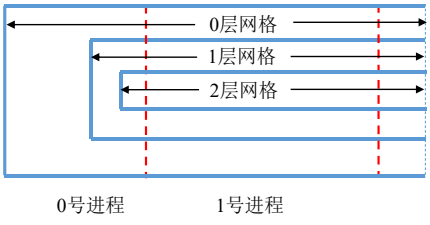
格划分给若干个进程. 在计算中,若干个进程是相互影响的,但在本节中仅考虑前两个进程的时间开销情况. 由于每个进程的时间开销主要受到负载量的影响,通过自定义划分保证每个进程的计算负载基本一致. 因此,对于前两个进程的时间开销分析如图8(*b*)所示. 由于计算负载通过网格点和权重的乘积表示,计算负载相同即保证两个1层网格点数与一个0层网格点数之和相等. 由此可见,当两层网格的每个进程负载量一致时,每个进程的时间开销理论上一致,两层网格间的计算无时序性.



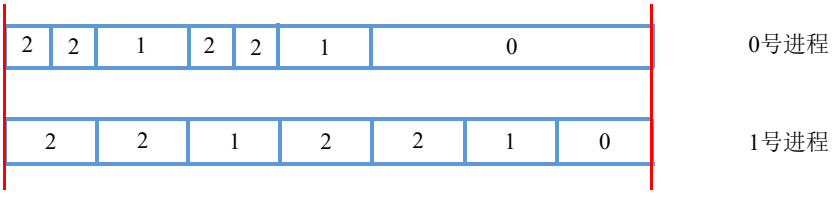
(*a*) 网格点划分,红色虚线表示进程分割线　　(*b*) 时间开销、矩形块长度表示时间开销长度

图8　两层网格中的网格点划分与时间开销(数字表示网格点层级数,c表示不同层级间的数据交换,红色实线表示进程间的通信过程)

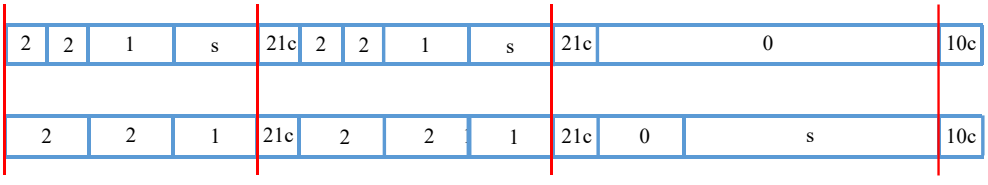
对于三层以上的网格系统,其时间开销原理与图9中分析的三层网格系统相同. 在图9(*a*)中,通过同样的方法将整体任务划分给若干个进程,同时使得每个进程的负载量相等,其时间开销如图9(*b*)所示. 即不同进程的4个2层网格点数、2个1层网格点数和1个0层网格点数的总和相等. 然而将图9(*b*)中的时间块返回到计算过程中,如图9(*c*)所示,对于每个进程都产生了额外的等待时间. 由于自定义划分的规则是保证所有进程的负载量相同,而不是保证每个进程在各层网格内的负载量相同. 当进行不同层的网格数据交换时,局部的负载量不同导致并行进程之间存在时序性. 因此,由于"缓冲区"的存在,平衡每个进程的总体负载量不能得到很好的效果. 而是需要通过平衡每个进程在每个网格尺度上的负载量,降低时序性的影响,从而达到良好的负载均衡效果.



(*a*) 网格点划分,红色虚线是进程分割线



(*b*) 计算负载相同时的时间开销状态



(*c*) 计算过程中的时间开销

图9　三层网格的网格点划分、计算负载相同时的时间开销状态与计算过程中的时间开销(数字表示网格点层级数,红色实线表示进程间的通信过程, c表示不同层级间的数据交换,c前数字表示参与交换的层级,s表示等待时间)

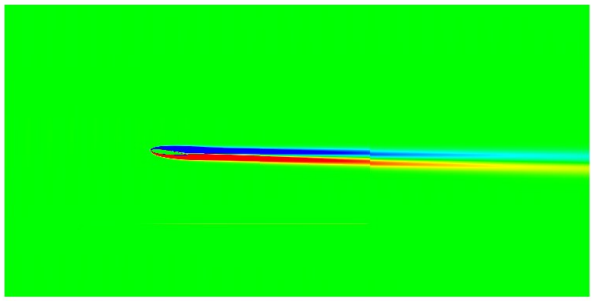
5.2 多层网格中的负载均衡

在5.1节中得到结论:两层网格间无时序性,三层及以上的网格存在时序性.如果仅考虑进程间的负载量相等,时序性的存在会产生额外的等待时间,从而增加总体的计算时间.为了解决这个问题,并不需要复杂的剖分算法来划分网格负载,而是需要将自定义划分中的进程间总负载量相等,换成进程在每个网格尺度上的负载量相等.在前一节中通过理论分析得出这个结论,在本节中通过二维翼型绕流的计算来分析验证这个结论.

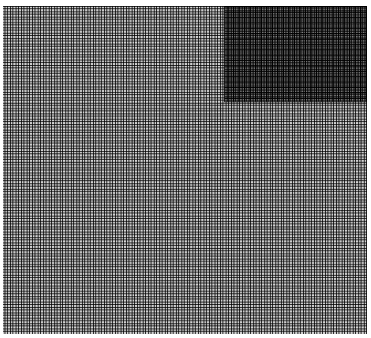
二维翼型绕流的计算中,使用不同的网格参数计算相同工况下的绕流问题.整体计算域的大小为$16L\times 8L$,L为二维翼型弦长,翼型迎角为7°,U为均匀来流,v为粘性系数,雷诺数Re为1 000由$\mathrm{Re}=UL/v$定义.总体计算步数为10万步,计算平台与4.1节的相同,均采用128核进行满负载运算.在图10中给出四层网格下的涡量模式和全部的压力系数结果.时间开销和网格参数如表2所示.其中图10(a)中的涡量模式与文献[23]中相同工况下的结果一致,图10(c)中网格层数越多,二维翼型的压力系数与文献越吻合.

为了验证本文在多层网格并行计算中提出的优化方法,使用不同的网格点划分方式对二维翼型绕流进行计算,对比一维均匀划分、总负载量均匀划分和各层负载量均匀划分.其中,一维均匀划分为常规划分方式,在一维划分中沿x轴正向进行均匀切割.总负载量均匀划分在均匀划分的基础上,通过平移网格的分割线从而使得每个进程的计算负载接近,其中负载的定义在5.1节给出.各层负载量均匀划分不是保证每个进程总负载相同,而是在每一层网格中,计算负载都是接近相同.这些不同的网格划分方式,采用本文提出的任意网格划分下的相同程序进行计算,从而排除了程序设计的干扰.三种划分方式的示意图如图7所示,其中图7中给出的二层网格的状态,其余层数的网格设置与之类似.

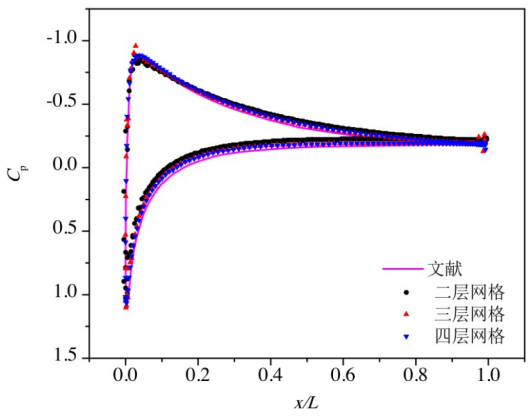
在图11(a)中沿x轴向进行一维均匀划分,用于表示网格点进程归属的色块大小基本一致.虽然这种方式最为简便,但会造成每个进程划分的计算负载相差较大.很容易地看到,当色块覆盖的区域面积相等时,区域内网格尺寸越小,区域内网格点越多,同时网格点在计算负载中所占权重越大.为了平衡所有进程的负载量,通过调整进程分割线,可以缩小进程间的负载量差距.在图11(b)中可以看到中间的色块偏小,两端的色块偏大,然而这种方法只能缩小进程间的总负载量差距.在图11(c)中对每一层分别进行划分,做到每个进程在每一层的负载量差距最小.三种划分方式中不同进程的负载量如图11(d)示,在多层网格结构中,均匀划分会造成进程间存在极大的负载量差距,而后两种划分方法都可以很大程度上降低这种差距.



(a) 二维翼型绕流的涡量模式



(b) 网格局部放大图



(c) 三种不同网格参数下的压力系数分布

图10 四层网格下的翼型绕流的涡量模式、网格局部放大图与三种不同网格参数下的压力系数分布

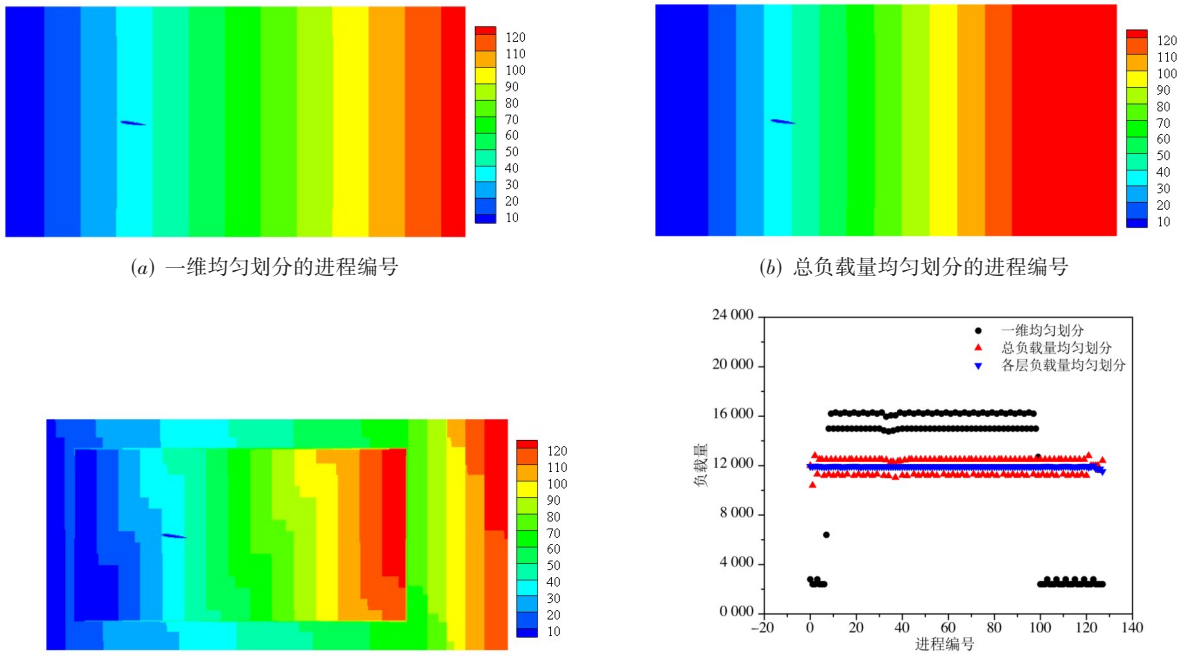
在图11中对两层网格下的二维翼型绕流使用三种不同的方式进行网格点划分,同时分析了三种方式下的不同进程负载量情况.对于三种不同的划分方式,记录运算一万步的时间开销,如表3所示.在表3中,总体时间开销统计LBM中主体程序块的运算时间,主要包含碰撞、迁移、曲线插值、宏观量计算和边界条件修正.而运算时间开销分别独立统计以上的每个运算块的时间开销后取和值.两个时间开销的差可以表示进程的

表2　不同网格参数下的网格量与时间开销

网格层数量	网格点数量/万		时间开销/s
二层网格	0号层	15.1	972
	1号层	68.8	
	合计	83.9	
三层网格	0号层	15.1	3 816
	1号层	47.5	
	2号层	134.4	
	合计	195	
四层网格	0号层	15.1	17 352
	1号层	32.4	
	2号层	69.9	
	3号层	311	
	合计	428	

等待时间,采用这种方法来避免数据通信时的进程同步带来的偏差.

在表3中,当网格为两层网格时,一维均匀划分的负载量在不同进程中出现明显的差距,最大负载量是最小负载量的6倍以上,时间开销在三种划分中处于最大值. 而当负载量的差距减小时,时间开销逐步减小. 说明此时进程的时间开销主要由负载量控制,总体的时间开销取决于最大负载量对应进程的时间开销. 在三层及以上的多层网格中,考虑负载均衡的划分方式的负载量相差较小,然而总负载量均匀划分的时间开销大于各层负载量均匀划分. 总负载量均匀划分的运算时间少于一维均匀划分的运算时间,但等待时间占比却出现明显的增大,而这种现象在多层网格加密越多越明显. 这种结果证实了多层网格中时序性的存在,即总负载量均匀划分可以降低最大时间开销,但影响了进程间的同步计算性能,而各层负载量均匀划分做到了每一层的进程负载量平衡,因此可以避免时序性的影响,从而减小了进程间等待的时间.



(*a*) 一维均匀划分的进程编号　(*b*) 总负载量均匀划分的进程编号

(*c*) 各层负载量均匀划分的进程编号　(*d*) 三种划分方式的进程负载量分布

图11　二层网格下的不同划分方式与不同进程的负载量分布

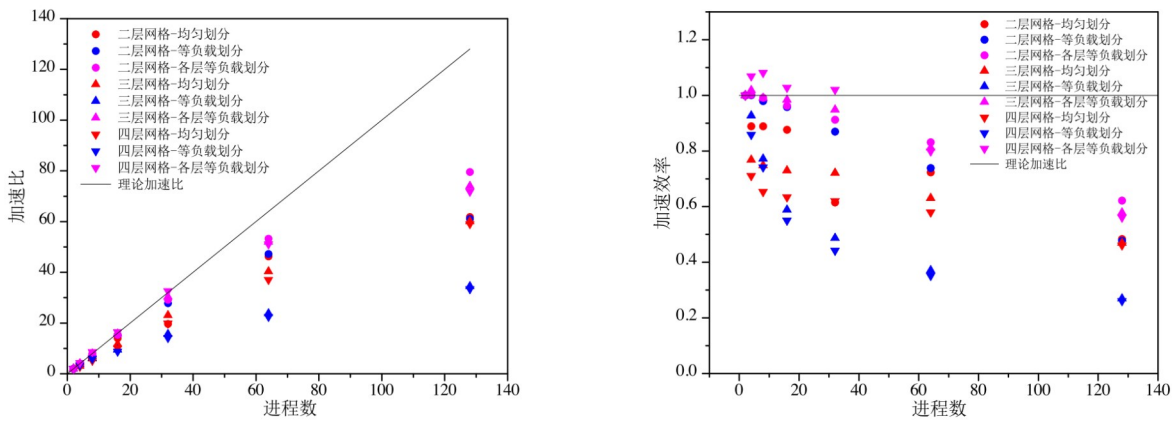
5.3　网格划分优化与并行性能

在单层网格的流场计算中,通过调整不同进程间的负载量,可以降低整体的时间开销. 在多层网格的计算中,由于多层网格中不同网格的计算时序性的存在,总体的负载均衡不能保证各层网格中的负载均衡. 而在各层等负载划分中,可以保证各层网格中的负载均衡效果,从而得到很好的优化效果. 为了评估算法的并行性能,我们通常会考虑算法的两种可扩展性,即强可扩展性和弱可扩展性. 对于强可扩展性,指算法的效率恒定;对于弱可扩展性,指问题规模以一定速率增大,效率不会随着线程数量的增大而降低. 在图12中给出三种网格划分算法的强可扩展性.

从图12中可以看到,在三种不同网格结构中,各层等负载划分的并行性能最优. 在128核并行平台的运算中,半负载并行效率达到80%左右,满负载并行效率达到60%左右. 进一步分析测试并行性能过程中的详

表3 不同划分方式下不同网格参数下的负载量和时间开销

网格划分方式	负载量(最小/最大)	时间开销(总)/s	时间开销(运算)/s	等待时间(占比)/%
一维均匀划分(二层)	2 400/16 302	182.52	158.95	12.9
总负载量均匀划分(二层)	10 396/12 796	128.37	112.82	12.1
各层负载量均匀划分(二层)	11 537/11 977	**108.23**	**103.61**	**4.3**
一维均匀划分(三层)	2 400/95 266	774.83	649.13	16.2
总负载量均匀划分(三层)	48 848/55 372	1 064.35	552.81	48.1
各层负载量均匀划分(三层)	50 751/52 385	**407.68**	**380.94**	**6.6**
一维均匀划分(四层)	2 400/568 680	3 895.86	3 409.93	12.5
总负载量均匀划分(四层)	215 288/230 416	4 168.66	2 199.20	47.2
各层负载量均匀划分(四层)	219 243/223 057	**1 692.02**	**1 623.88**	**4.1**



(*a*) 加速比 (*b*) 并行效率

图12 不同网格结构下的不同划分方式的加速比和并行效率

细数据,如表4所示. 当进程数量较少时,进程间的时序性影响相对较小,时间开销主要受到计算负载影响. 因此,相比与均匀划分,等负载划分存在一定的优化效果. 总的来说,各层等负载划分在全部测试中都是保持最佳的优化效果. 为了进一步测试最佳网格划分策略下的弱可扩展性,对网格结构进行了一些调整,使得各层网格的负载量随着进程数量同步进行增加. 多层网格为三层网格结构,测试参数和结果如表5所示,在128进程环境下,弱可扩展性达到82.78%. 综合表4和表5的结果可以看到,本文方法在更多层数的多层网格和更多核数时具有更好的效果,因此更加适用于网格规模大和计算核数多的并行计算.

表4 并行性能测试过程中的详细时间开销 单位:s

进程数	2	4	8	16	32	64	128
均匀划分(二层)	230.26	129.50	64.77	32.85	23.40	9.94	7.45
等负载划分(二层)	196.43	98.19	50.17	25.63	14.12	8.31	6.43
各层等负载划分(二层)	201.54	100.57	50.85	26.17	13.81	7.57	**5.07**
均匀划分(三层)	1 018.59	663.14	340.46	174.40	88.32	50.48	33.90
等负载划分(三层)	743.71	401.07	240.98	157.97	95.43	62.64	43.13
各层等负载划分(三层)	6 94.32	340.86	175.28	88.23	45.76	26.73	**18.77**
均匀划分(四层)	5 300.50	3 729.02	2 027.44	1 044.36	533.93	285.77	179.46
等负载划分(四层)	3 041.86	1 771.83	1 025.21	690.95	429.81	270.02	181.78
各层等负载划分(四层)	3 072.28	1 537.05	769.96	393.60	198.11	120.19	**85.48**

表5　最佳网格划分策略下的弱可扩展性

进程数量	0层负载量	1层负载量	2层负载量	时间开销/s	弱可扩展性
2	3 324	8 320	24 236	15.96	100%
8	12 896	31 304	86 683	16.3	97.91%
32	50 796	122 604	332 877	16.11	99.07%
128	201 596	485 204	1 304 062	19.28	82.78%

6　结论

本文主要从网格划分与程序实现两方面,研究了单层网格和多层网格LBM在并行计算中的负载均衡性能及其优化. 在网格划分方面,单层网格中每个进程的时间开销主要受到负载量影响,负载划分均匀即可实现负载均衡;多层网格中每个进程的时间开销需要考虑多尺度网格的时序性,因此每个网格尺度上均保持负载划分均匀,即可实现良好的负载均衡效果. 在程序实现方面,将程序实现与网格划分独立开来,通过全网格点的进程标记,来实现任意网格划分下的并行计算. 这种处理可排除不同的程序实现对时间开销的影响,有助于更好地研究精确有效的网格划分策略. 在流场的计算验证中,通过不同网格划分下二维血管计算,除了验证LBM计算的准确性的同时,还说明了网格划分不影响流场结果,同时验证了负载量与同步率的相关关系. 通过不同多层网格结构下的二维翼型绕流计算,除了进一步验证了网格划分与流场结果互不影响,还验证了多层网格之间存在的时序性. 同时说明了各层网格下的均匀负载划分的时间开销更少. 在128核高性能计算平台中测试并行性能,当使用128个进程进行满负载运算时,强可扩展性达到60%左右,弱可扩展性达到82.78%.

参考文献

[1] BOON J P. The lattice boltzmann equation for fluid dynamics and beyond[J]. European Journal of Mechanics - B, 2003, 22(1): 101.

[2] SHEIKHOLESLAMI M, GANJI D D. Entropy generation of nanofluid in presence of magnetic field using lattice Boltzmann method[J]. Physica A Statistical Mechanics and Its Applications, 2015, 417: 273-286.

[3] QIAN Y H, D´HUMIÈRES D, LALLEMAND P. Lattice BGK models for navier-stokes equation[J]. Europhysics Letters, 1992, 17 (6): 479-484.

[4] CHEN S, CHEN H, MARTNEZ D, et al. Lattice Boltzmann model for simulation of magnetohydrodynamics[J]. Physical Review Letters, 1991, 67(27): 3776-3779.

[5] HOU S, STERLING J, CHEN S, et al. A lattice Boltzmann subgrid model for high Reynolds number flows[EB/OL]. (1994-01-28) [2023-02-01]. http://arxiv. org/abs/comp-gas/9401004

[6] 张凌明, 赵悦, 李鹏程, 等. 基于局部注意力机制的三维牙齿模型分割网络[J]. 电子学报, 2022, 50(3): 681-690.
ZHANG L M, ZHAO Y, LI P C, et al. Dental model segmentation network based on local attention mechanism[J]. Acta Electronica Sinica, 2022, 50(3): 681-690. (in Chinese)

[7] WELLEIN G, ZEISER T, HAGER G, et al. On the single processor performance of simple lattice Boltzmann kernels [J]. Computers & Fluids, 2006, 35(8/9): 910-919.

[8] KÖRNER C, POHL T, RÜDE U, et al. Parallel lattice Boltzmann methods for CFD applications[C]//Numerical Solution of Partial Differential Equations on Parallel Computers. Berlin: Springer, 2006, 439-466.

[9] DELANAYE M, AFTOSMIS M, BERGER M, et al. Automatic hybrid-Cartesian grid generation for high-Reynolds number flows around complex geometries[C]//Proceedings of the 37th Aerospace Sciences Meeting and Exhibit. Reston: AIAA, 1999: AIAA1999-777.

[10] FILIPPOVA O, HÄNEL D. Grid refinement for lattice-BGK models[J]. Journal of Computational Physics, 1998, 147(1): 219-228.

[11] TÖLKE J, KRAFCZYK M, RANK E. A multigrid-solver for the discrete boltzmann equation[J]. Journal of Statistical Physics, 2002, 107(1): 573-591.

[12] YU D Z, MEI R W, SHYY W. A multi-block lattice Boltzmann method for viscous fluid flows[J]. International Journal for Numerical Methods in Fluids, 2002, 39(2): 99-120.

[13] 王相海, 宋若曦, 曲思洁, 等. 图像多尺度几何分析域隐马尔可夫树模型研究进展[J]. 电子学报, 2022, 50(1): 238-249.
WANG X H, SONG R X, QU S J, et al. Advance in multiscale geometric analysis image hidden markov tree model[J]. Acta Electronica Sinica, 2022, 50(1): 238-249. (in Chinese)

[14] NEUMANN P, NECKEL T. A dynamic mesh refinement technique for Lattice Boltzmann simulations on octree-like grids[J]. Computational Mechanics, 2013, 51(2): 237-253.

[15] HASERT M, MASILAMANI K, ZIMNY S, et al. Complex fluid simulations with the parallel tree-based Lattice Boltzmann solver Musubi[J]. Journal of Computational Science, 2014, 5(5): 784-794.

[16] 袁海英, 曾智勇, 成君鹏. 面向灵活并行度的稀疏卷积神经网络加速器[J]. 电子学报, 2022, 50(8): 1811-1818.
YUAN H Y, ZENG Z Y, CHENG J P. A sparsity-aware convolutional neural network accelerator with flexible parallelism[J]. Acta Electronica Sinica, 2022, 50(8): 1811-

1818. (in Chinese)
[17] LIU Z X, RUAN J, SONG W, et al. Parallel scheme for multi-layer refinement non-uniform grid lattice boltzmann method based on load balancing[J]. Energies, 2022, 15 (21): 7884.
[18] LIU Z X, CHEN R L, XU L. Parallel unstructured finite volume lattice Boltzmann method for high-speed viscid compressible flows[J]. International Journal of Modern Physics C, 2022, 33(5): 2250066.
[19] GUO Z L, ZHENG C G, SHI B C. Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice Boltzmann method[J]. Chinese Physics, 2002, 11(4): 366-374.
[20] YU D Z, MEI R W, SHYY W. A unified boundary treatment in lattice Boltzmann method[C]//Proceedings of the 41st Aerospace Sciences Meeting and Exhibit. Reston: AIAA, 2003: AIAA2003-953.
[21] QI J X, KLIMACH H, ROLLER S. Implementation of the compact interpolation within the octree based Lattice Boltzmann solver Musubi[J]. Computers & Mathematics with Applications, 2019, 78(4): 1131-1141.
[22] GE S, XIE J, WANG H Y, et al. Large-scale lattice Boltzmann method simulations of stenosed carotid bifurcation[C]//2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Piscataway: IEEE, 2019: 1076 - 1084.
[23] KURTULUS D F. On the unsteady behavior of the flow around NACA 0012 airfoil with steady external conditions at Re=1000[J]. International Journal of Micro Air Vehicles, 2015, 7(3): 301-326.

作者简介

何 鹏 男,1994年6月出生于湖北省荆州市. 上海大学计算机工程与科学学院博士研究生,主要研究方向为高性能计算、计算流体力学.
E-mail: 2359597269@qq.com

王良军 男,1991年1月出生于安徽省阜阳市. 现就职于上海大学,研究方向为高性能计算、计算流体力学.
E-mail: shu_wlj@shu.edu.cn

张 武 男,1957年11月出生于江西省武宁县,上海大学力学与工程科学学院教授、博士生导师,研究方向为计算流体力学、并行计算算法与软件.
E-mail: wzhang@shu.edu.cn

朱文浩 男,1979年9月出生于江苏省昆山市,上海大学计算机工程与科学学院教授、博士生导师,研究方向包括人工智能、文本处理、大数据应用等.
E-mail: whzhu@shu.edu.cn