非结构网格并行计算预处理方法研究

刘 鑫 陆林生 陈德训
(江南计算技术研究所 无锡 214083)

摘 要 非结构网格预处理方法是非结构网格CFD并行计算的关键技术之一。提出基于缓冲数据结构的快速搜索算法来建立全局网格单元邻接关系图,算法复杂度低,能够显著降低非结构网格预处理的存储需求;在提高核心计算访存命中率方面,提出网格单元重排序算法,该算法能够提高核心计算效率,并通用于各种非结构网格问题。实验结果表明,在用于大网格量的复杂计算区域时该非结构网格预处理技术仍能得到较理想的结果。

关键词 非结构网格,CFD并行计算,预处理,快速搜索,重排序
中图法分类号 TP301 **文献标识码** A

Research on Pre-processing Methods of Unstructured Grids

LIU Xin LU Lin-sheng CHEN De-xun
(Jiangnan Institute of Computing Technology, Wuxi 214083, China)

Abstract The pre-processing methods of unstructured grids are one of the important technologies for unstructured grids CFD parallel computing. The paper supplied a new efficient and robust fast search algorithm to build the relationship graph of the global unstructured cells, which is based on buffer data structure and can be easily implemented with low complexity. And the paper brought forward the reordering algorithm to deal with the out-of-order problem brought by unstructured grids, which can improve the computing efficiency and can be used in all kinds of unstructured grids. Experiment results show that even in the case of complicated areas of large grids number, the pre-processing methods can get good performance.

Keywords Unstructured grids, CFD parallel computing, Pre-processing, Fast Search Algorithm, Reordering

1 引言

在科学计算领域中,结构网格[1]由于技术成熟、流场计算精度高、边界处理能力强等优点而大量应用于科学计算。但随着CFD领域的基础研究越来越深入,关键技术的复杂性和难度也越来越大,设计要求日益精细和完善,结构网格已经不能完全满足复杂拓扑结构问题的数值模拟要求,需要采用非结构网格与结构网格互相补充,才能更精确地模拟复杂外形和结构的飞行器问题。近年来,非结构网格在计算流体力学中的应用越来越广,它由于不受网格结点结构性的限制,易于控制网格单元的大小、形状及网格结点的位置,因此比结构网格具有更大的灵活性,对复杂外形的适应能力非常强。同时,非结构网格还适用于基于流场解的网格自适应技术,从而有效提高流场求解精度。

非结构网格并行计算预处理需要解决的问题主要有非结构网格的负载平衡问题、降低存储需求问题、网格单元重排序问题等。非结构网格的负载平衡当前使用较多的是基于图的多层划分方法[2],其中METIS[3]的发展比较成熟,已经广泛应用于CFX[4]、FLUENT[5]等CFD计算软件的预处理过程中。其次,在非结构网格预处理过程中,需要建立全局网格单元的相互关系,存储量大大增加,如何在保证计算效率的前提下减少预处理过程中的存储量需求,是非结构网格大规模并行计算的关键问题之一。此外,由于非结构网格舍去了网格结点的结构性限制从而在数据存储上造成无序性,数据存储量大,计算效率低,收敛性和稳定性也较差,网格和节点的编号极没有规律,使得很多相邻网格之间及相邻节点之间的编号跨度很大,从而其流场参数在内存中的存储极没有规律,计算过程中调用各网格的信息需花费很长时间,影响计算效率,因此需要对原始网格单元进行重排序以提高访存命中率,同时改善求解线化方程时的矩阵品质、加速收敛。

本文采用较成熟的非结构网格分块软件METIS进行负载平衡和任务划分;在构造METIS输入参数即建立各网格单元相邻关系图时,为解决预处理内存需求过大问题,本文提出一种基于缓冲结构的快速搜索算法来建立全局网格单元邻接关系图;在METIS分块完成以后,为提高核心计算访存命中率,采用一种新的网格重排序算法,使得相邻网格之间及相邻节点间的编号跨度尽可能小,从而减少计算中调用各网格信息的时间花费,提高计算效率。从典型算例的计算结果可看出,该非结构网格预处理方法对提高计算效率、降低内存需求是比较有效的。

到稿日期:2011-04-15 返修日期:2011-07-26 本文受863计划(2010AA012301)资助。
刘 鑫(1979-),女,博士,副研究员,主要研究方向为并行算法及应用,E-mail:yyylx@263.net;陆林生(1942-),男,教授,博士生导师,主要研究方向为并行算法与并行识别;陈德训(1972-),男,高工,主要研究方向为并行算法及应用。

2 非结构网格并行计算预处理方案

与结构网格类似，根据该类问题的基本计算方法[1]，其并行计算程序应遵循分块并行求解和边缘通信的准则。流场分为多个计算区域后，需要保证各块网格在公共边界上的网格重合，在求解过程中，因无粘对流项的计算采用 MUSCL 插值方法，粘性项的计算利用改进的高斯定理，所以并行计算时各计算区域的边界应互相重叠两层网格，计算中需交换重叠部分网格上的物理量值和守恒变量的隐式增量。实际在并行计算实现时，读入流场计算参数后，应根据负载均衡的原则将整个流场计算分配给不同的处理器，由不同的处理器负责自己的流场计算，并在适当的时候与相关处理器互相通信，传递虚边界上的信息，直至最后流场收敛。

非结构网格预处理过程主要分为如下几步(见图 1)：1)读入网格文件；2)根据输入网格文件信息，以网格单元为顶点、相邻网格面为边，建立全局网格单元邻接关系图作为 METIS 分块的输入；3)调用 METIS 函数库进行分块；4)根据 METIS 分块结果设置进程内网格单元的基本信息；5)为提高 Cache 命中率、改善求解线化方程时的矩阵品质，需要对排序后网格单元进行重排序；6)对排序后网格单元设置进程间通信关系索引。为降低存储需求，仅在 1)、2)、3)、4)步保存全局网格单元邻接关系图信息，4)、5)、6)步均在每进程的私有数据空间内完成。

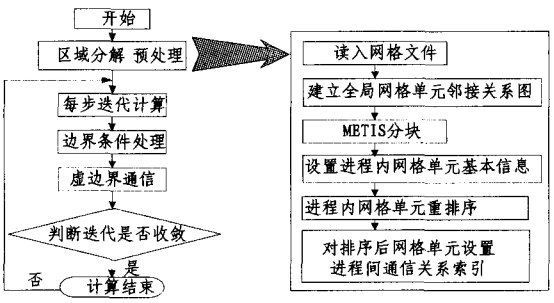

图 1 非结构网格并行计算预处理流程图

3 建立全局网格单元邻接关系图的快速算法

3.1 降低存储需求的网格单元缓冲

为建立全局网格单元邻接图，对于千万网格规模的输入数据，仅存储全局网格单元信息和全局网格单元面信息，其存储量就将超过 4GB。为减少建立全局网格单元邻接关系图所需要的全局网格单元和面信息存储需求，引入网格单元缓冲数据结构，用于存放部分网格单元和面信息，对全局网格单元进行分批处理，具体步骤如下：

每个网格单元邻居数初始化设置为 0；

对网格单元进行循环；

若网格单元邻居数为 0，则将该网格单元装入网格单元缓冲中；如果该网格单元是物理边界单元，则邻居数设置为该网格单元的物理边界面数；

若缓冲满，则对网格单元缓冲内网格单元求相交，根据相交结果设置网格单元邻居数，若该网格单元邻居数等于网格单元面数，则将该网格单元淘汰，清空网格单元缓存中该网格单元的信息；如果是最后一次网格单元缓冲处理，为提高求相交效率，需要将后端所有非零单元集中放到前面。

所有网格单元处理完毕后退出循环。

3.2 网格单元求相交快速算法

判断两网格单元是否相邻，可通过输入网格信息得到该网格单元由哪几个网格单元面组成，若存在相同的网格单元面，则这两个网格单元为相邻网格单元。因此，网格单元求相交的计算量主要集中在网格面搜索中。在具体实现过程中，对 Hash 搜索算法的基本思想进行引申，采用构造最小网格点索引的办法完成网格面的快速搜索。最小网格点索引表的大小为全局网格点数，索引表中存放指向第一个以该网格点为起点的网格单元面序号 *p_face*、所有以该网格点为起点的网格单元面个数 *n_face* 两个数据项；同时原网格面信息中新加 next 指针，用来存储与其具有相同最小序号网格点的下一个网格面。如果对新加的小面按最小网格点序号进行查询，最多比较 *n_face* 次就可以查询到是否与已有网格面重复。具体实现步骤如下：

对网格单元进行循环

根据网格单元类型得到组成网格单元面的网格点列表，取每个面的最小序号点排在首位，其余点按原顺序循环放在该点后面；

网格单元信息进行赋值；

对网格单元面进行循环

在已有面中进行搜索，如果有则更新相邻关系，没有则建立新的网格单元面信息。

结束网格面循环

结束网格单元循环

4 网格单元重排序算法

4.1 问题提出和部分基本定义

基于第 3 节建立的全局网格单元邻接关系图，调用 METIS 进行分块，根据分块结果完成进程内网格单元基本信息的设置后，为提高 Cache 命中率、改善求解线化方程时的矩阵品质，还需要对排序后的网格单元进行重排序。块内网格单元及网格面重排序算法主要用于线化方程求解的索引，使其能实现由边界网格单元出发的上推和下推，同时改善矩阵的品质；此外，重排序算法还可以使小块内的网格单元局部有序，从而进行分层处理，提高 Cache 命中率。为更好阐明算法原理，首先引入一些定义。

定义 1(全局标号和局部标号) 经过自动分块后赋予小块包含的网格单元体(cell)、网格单元面(face)和网格点(node)的标号成为全局标号，由重排序得到的标号成为局部标号。

定义 2(近边界、近物理边界、近通信边界、近混合边界网格单元) 如果网格单元至少有一个网格面是物理边界面，则称其是近物理边界网格单元；如果网格单元至少有一个网格面是通信边界面，则称其是近通信边界网格单元；如果网格单元既是近物理边界网格单元，又是近通信边界面网格单元，则称其是近混合边界网格单元。上述这些网格单元统称为近边界网格单元。

定义 3(边界、物理边界、通信边界网格单元) 由近物理边界网格单元向边界外开拓的网格单元称为物理边界网格单元；由近通信边界网格单元向通信边界外开拓的网格单元称为通信边界网格单元。上述这些网格单元统称为边界网格单元。

定义 4(内部网格单元) 如果网格单元既不包含物理边界面,又不包含通信边界面,则称其是内部网格单元。

4.2 算法步骤

网格单元重排序算法形式化描述:

步骤 1 对小块原始的网格单元分层,标志 global_level,分层方法如下:

步骤 1.1 根据邻居关系找出连通的边界单元集合,如果有多个集合,则取单元数最多的 1 个集合,将其中的网格单元层号记为 global_level=1,并将这些网格单元放入搜索数组中;

步骤 1.2 对于搜索数组中的每个网格单元,根据邻居关系找出尚未标志 global_level 的相邻单元集合,将其中的网格单元层号记为 global_level+1;

步骤 1.3 如果所有的网格单元都标记了层号,算法结束,否则,将这些网格单元放入搜索数组中,转步骤 1.2。

步骤 2 选择重排序的初始单元

步骤 2.1 在 global_level=2 的集合中(即近边界单元),选取其相邻单元中边界单元数最多的那个单元作为初始单元;

步骤 2.2 找出初始单元的相邻边界单元,将相邻边界单元和初始单元依次放入缓冲。将缓冲中的单元依次写入 CellIndex 中,置 level=1;

步骤 2.3 将缓冲中的单元依次写入工作队列 work_queue 中。

步骤 3 对工作队列 work_queue 操作,进行重排序

步骤 3.1 由工作队列取出 1 个单元作为搜索单元,找出其相邻边界单元,根据相邻边界单元的 global_level 由小到大依次放入缓冲;

步骤 3.2 将缓冲中的单元依次写入 CellIndex 中,level 是搜索单元的 level+1;

步骤 3.3 将缓冲中的单元依次写入工作队列 work_queue 中;

步骤 4 如果队列空,重排序结束,否则,转步骤 3。

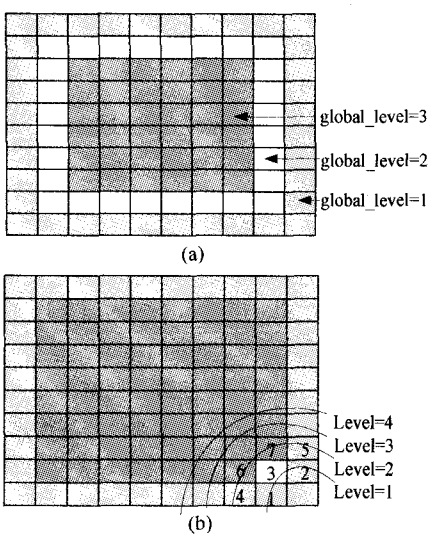

图 2 网格单元重排序算法示意图

如图 2(a)所示,经过步骤 1 的操作,global_level=1 的单元一定是边界单元,global_level=2 的单元一定是近边界单元,某个网格单元的邻居单元一定处于相同层和相邻层内。步骤 2 的主要工作原理如图 2(b)所示,首先选取网格单元 3 为初始单元,其相邻边界单元 1、2 与其一起计入 level=1 的网格单元集合中,并按序进入工作队列,同理 1 的相邻单元 4、2 的相邻单元 5、3 的相邻单元 6 和 7 一起计入 level=2 的网格单元集合,按序进入工作队列中,直至所有网格单元 level 设置完成。重排序后的网格单元集合有以下特性:(1) level 小的网格单元集合的标号一定小于 level 大的网格单元集合的标号;(2) 利用 level 由小到大或由大到小推进,类似于结构网格的对角推进,而且也由边界单元出发;(3) 如果根据一定原则,在排序后网格单元集合的基础上将相邻层次的网格单元组织在一起进行计算,将有利于 Cache 命中率的提高。

5 实验结果

5.1 基于缓冲数据结构的快速搜索算法效率实验

在主频为 2.93G 的 Intel X5670 计算节点上进行了非结构网格预处理过程中建图部分的时间效率测试,因二维算例计算时间太短,所以统一采用三维算例。表 1 为三维单机翼、发动机整机模型、飞行器整机模型的非结构网格预处理过程占用内存(包括读文件、建图以及 METIS 分块等共同的占用内存量)以及建图墙钟时间的具体实验结果。

表 1 部分三维算例预处理效率

典型算例	网格数(万)	预处理占用内存(M)	建图墙钟时间(秒)
三维单机翼	29.49	91.10	0.15
发动机整机模型	55.05	161.20	0.27
飞行器整机模型	1054.35	2080.10	4.61

5.2 重排序计算结果

采用湍流平板和 M6 机翼算例验证重排序效果,分别采用 2、16 并行规模进行计算。由表 2 可以看出,重排序算法使小块内的网格单元局部有序进行分层处理,能够提高 Cache 命中率,降低每步计算时间;另外,重排序的计算结果与原网格的计算结果基本一致,但同时也注意到,从残差收敛速度角度看,重排序算法没有起到改善求解线化方程时的矩阵品质、加速收敛的效果,还需要做进一步的深入研究和更广泛的算例测试。

表 2 部分三维算例重排序 100 步计算时间

典型算例	排序后 100 步计算时间	排序前 100 步计算时间
M6 机翼	13.11	15.30
湍流平板	13.52	15.45

结束语 在非结构网格预处理过程中,提出一种基于缓冲数据结构的快速搜索算法来建立全局网格单元邻接关系图,算法复杂度低,能够显著降低非结构网格预处理的存储需求;在提高核心计算访存命中率方面,提出网格单元重排序算法,算法健壮性较好,能够提高核心计算效率,并通用于各种非结构网格问题。实验结果表明,该非结构网格预处理技术能得到较理想的结果。同时也注意到,从湍流平板的残差收敛结果来看,重排序算法没有起到改善求解线化方程的矩阵品质、加速收敛的效果,还需要做进一步的深入研究和更广泛的算例测试。

参 考 文 献

[1] 刘鑫. 面向化学非平衡流的 CFD 并行计算技术和大规模并行计算平台研究[D]. 郑州:解放军信息工程大学,2006:69-73

[2] Monien B, Preis R, Diekmann R. Quality matching and local improvement for multilevel graph-partitioning[J]. Parallel Computing, 2000, 26(12): 1609-1634

[3] Karypis G, Kumar V. METIS: unstructured graph partitioning and sparse matrix ordering system[R]. Department of Computer Science, University of Minnesota, 1995

[4] ANSYS FLUENT [OL]. http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/ANSYS+FLUENT

[5] ANSYS CFX [OL]. http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/ANSYS+CFX

(上接第 285 页)

通过安全的身份鉴别后，系统将监控任务提交到全局生产者，然后由全局生产者向下派发监控任务并最终得到返回的监控信息，将其进行处理后返回给用户。图 2 和图 3 是普通用户获得的 JSMON 系统监控结果的截图。

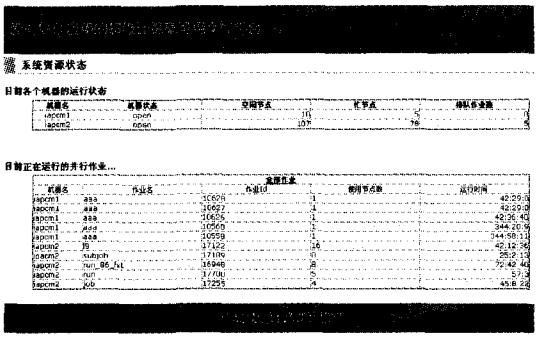

图 2 作业和资源运行状态

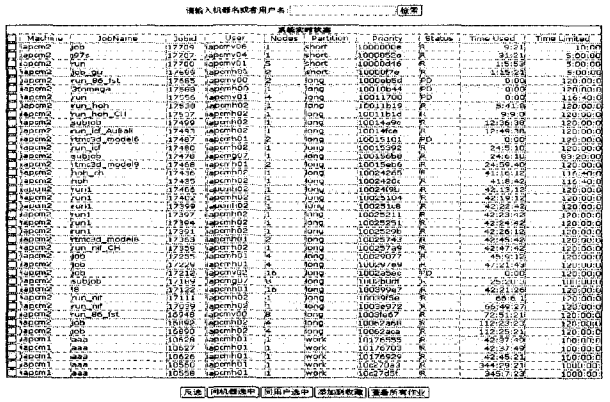

图 3 作业详细监控信息

4.3 性能评价

JSMON 获取所监控节点的资源运行状态和资源消耗情况。其对被监控节点的性能数据的采集是通过运行在节点上的资源采集 Client 来获得的，按照系统预定的周期，采集监控数据提交给生产者，以便上层应用使用。

由于采用了多线程对数据进行采集来控制传输时间，因此在采集部分和解析部分数据可以并行处理，而在网络条件较好的时候，传输也可以并行进行，这样数据的最大时间从之前的加和模式变成了最大模式，在对两台高性能计算机的监控测试的时候发现，数据监控时间开销从 0.2342s 下降到 0.1765s。

通过引入用户访问模式来减少整个监控系统的监控次数，以达到降低负载和网络带宽的目的，在下一步的工作中将对这部分工作进行测试。

结束语 监控系统对于高性能计算环境管理来说是至关重要的。通过监控可以及时发现产生故障的主机、网络设备、应用程序等，分析系统性能瓶颈，帮助用户在最短的时间内调整或恢复系统。基于 GMA 我们设计并实现了一套基于高性能计算环境的监控系统 JSMON，并投入使用。该监控系统具有如下特点：

1. 系统可移植性强。系统采用 Java 组件技术，使用通用的标准 XML 技术传输信息，使用 B/S 结构作为 Web 服务功能，使得信息的获取、产生、传输、显示标准化，从而具备了很好的跨平台性。

2. 监控数据的可定制性，根据用户定制作业监控信息，实现作业监控信息的筛选，用户得到自己最关心的数据信息。用户以及系统管理员可以根据作业监控系统提供的数据了解监控域内每个集群或各个主机节点的运行情况。

3. 系统具有方便可用性。作业监控系统针对系统管理员和用户提供不同的监视视图。

4. 系统轻量级。由于采用高性能计算机现有的监控接口，不需要重新配置一套监控服务，因此对于集群本身的监控负载最小。

5. 系统实时性好。由于园区局域网络传输速度快、数据传输稳定、处于同一管理域内以及相对安全等特点，较之互联网，监控系统需要考虑的因素相对简单，则实时性会较互联网的监控系统好。

下一步将对监控系统的性能和安全性做进一步的研究，使得监控系统更加完善。

参 考 文 献

[1] Blaise Barney. Introduction to Livermore Comput- ing Resources. [EB/OL]. https://computing.llnl.gov/tutorials/lc_resources/, 2011-08

[2] Network Weather Service[EB/OL]. http://nws.cs.ucsb.edu

[3] GT 4.0 Index Service: User's Guide [EB/OL]. http://www-unix.globus.org/toolkit/docs/4.0/info/index/user-index.html

[4] Buyya R. PARMON: a portable and scalable monitoring system for clusters[J]. Software Practice and Experience, 2000, 30(7): 723-739

[5] Sottile M, Minnich R. Supermon: A Highspeed Cluster Monitoring System[C]//Proceedings of Cluster. 2002

[6] 王翠，刘方爱. 基于 GMA 的教育资源网格监控研究[J]. 计算机工程与设计，2009，30(16)：3914-3917

[7] 黄鑫，刘梅梅，朱巧明，等. 网格监控系统研究综述[J]. 计算机应用研究，2009，26(1)：1-3，18

[8] A Grid Monitoring Architecture [EB/OL]. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-2.pdf

[9] XML 系列教程[EB/OL]. http://www.w3school.com.cn/x.asp

[10] About RRD tool [EB/OL]. http://oss.oetiker.ch/rrdtool/, 2008

[11] 张宏海. 网格监控系统[J]. 超级计算通讯，2006，4(4)：43-55