基于 MPI 的不可压缩 N-S 方程并行计算方法的研究

李 宁[1],罗纪生[1,2]
LI Ning[1],LUO Ji-sheng[1,2]

1.天津大学 力学系,天津 300072
2.南开大学 天津大学 刘徽应用数学中心,天津 300072
1.Department of Mechanics,Tianjin University,Tianjin 300072,China
2.Liu-Hui Center of Applied Mathematics,Nankai University and Tianjin University,Tianjin 300072,China
E-mail:lining_tju@hotmail.com

LI Ning,LUO Ji-sheng.Research on parallel computing algorithm of incompressible N-S equation based on MPI. Computer Engineering and Applications,2007,43(9):8-10.

Abstract: For the present CFD,when we try to solve large-scale science computing problems,such as the N-S equation,we have to calculate a lot of things and it takes a lot of time to get all the calculations done.To solve the problem,a parallel computing algorithm called MPI,which includes the solution of tridiagonal matrix and SOR,is proposed.This method has been proved to be accurate and reliable,it can shorten the calculating time enormously and is well adaptable to large-scale scientific computing.
Key words: CFD;N-S equation;MPI;parallel computing

摘 要:在目前的计算流体力学问题中,当求解 N-S 方程等大型科学计算问题时,存在着计算量大、耗时长的问题,对此提出了一种 MPI 并行算法,其中包括并行求解三对角矩阵与超松弛迭代。通过实例验证,该方法准确、可靠,并且可以大大缩短计算时间,对于大型科学计算问题具有很好的适用性。
关键词:计算流体力学;N-S 方程;MPI;并行计算
文章编号:1002-8331(2007)09-0008-03 **文献标识码**:A **中图分类号**:O246

近几十年,随着计算机运行速度的快速提高,计算流体力学得到了蓬勃的发展,而这也促进了航空、航天技术的发展。显然,数值计算已经成为与理论分析和实验研究并列的研究流体流动问题的方法。然而,由于流体流动的复杂性,使得流场数值计算的工作量非常巨大,特别是采用空间模式,以至于利用目前运算速度最快的计算机仍然无法仔细模拟大多数工程流场。因此,计算流体力学工作者们一直在寻求更好的算法来大规模地提高计算速度。并行计算就是在这种背景下应运而生的。

基于 Linux 环境下的 MPI 并行计算,以其准确、可靠、高效的优点,备受计算流体力学工作者的青睐。本文所要阐述的问题即是采用 MPI 并行求解 N-S 方程。

1 MPI 并行简介

MPI(Message Passing Interface)是一个消息传递接口的标准[1],用于开发基于消息传递的并行程序,其目的是为用户提供一个实际可用的、可移植的、高效的和灵活的消息传递接口库。MPI 以语言独立的形式来定义这个接口库,这个定义不包含任何专用于某个特别的制造商、操作系统的特性。由于这个原因,MPI 被并行计算界广泛接受。基于消息传递的并行编程,是指用户必须显式地通过发送和接受消息来实现处理器之间的数据交换,在基于 MPI 编程模型中的计算由多个彼此通过调用的库函数进行消息收发通信的进程组成。MPI 是一个标准,MPICH 是 MPI 的一个实现,提供了模块化的函数可供调用,程序编写直观。MPI 以语言独立的形式来定义这个接口库,提供了与 C 与 Fortran 语言的绑定。

MPI 不是一个独立的自包含系统,而是建立在本地并行程序设计环境之上,其进程管理和 I/O 均由本地并行程序设计环境提供。MPI 实现主要包括 MPICH、LAM、CHIMP 几种。MPI 子集由 6 个基本函数组成:MPI 初始化、MPI 结束、获取进程的编号、获取制定通信域的进程数、消息发送和消息接受。MPI 支持以下四种通信模式:标准通信模式、缓冲通信模式、同步通信模式、就绪通信模式。

2 本文问题描述

不可压缩直接数值模拟的基本方程是 Navier-Stokes 方程、连续性方程,对方程进行适当的无量纲化后,其形式:

$$\frac{\partial \boldsymbol{u}}{\partial t}+(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}=-\nabla p+\frac{1}{\mathrm{Re}}\nabla^2\boldsymbol{u} \tag{1}$$

$$\nabla\cdot\boldsymbol{u}=0 \tag{2}$$

其中 $\boldsymbol{u}$ 是速度,P 为压力,Re 是 Reynolds 数,∇ 是 Hamilton 算子,∇^2 是 Laplace 算子。

在 z 方向采用 Fourier 谱展开,因此 z 方向为周期性边界

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.10232020)。
作者简介:李宁(1978-),男,博士研究生,主要研究方向为流体力学、并行计算;罗纪生(1954-),男,教授,博士生导师,主要研究方向为流体力学。

条件。x、y 方向采用紧致差分格式。因此,本文在 x、y 方向采用并行计算。时间的离散方法和空间的离散方法的详细情况见文献[2]。

针对非线性项、粘性项、散度和梯度,都需要求解三对角矩阵。设所要求解的三对角方程组为:$\boldsymbol{A}\cdot x=f$,其中,$\boldsymbol{A}$ 为三对角矩阵,并且是非奇异的。那么,如何并行求解三对角矩阵呢?

另外,针对压力函数 p,其满足的方程为 Helmholtz 方程:

$$\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}-b\cdot p=f$$

采用五阶中心差分格式离散上面的 Helmholtz 方程,并采用超松弛迭代法求解[3],可得:

$$P_{i,j}^{P+1}=(1-\omega)p_{i,j}^{p}+\omega\frac{\boldsymbol{A}(p_{i-1,j}^{p+1}+p_{i+1,j}^{p})+\boldsymbol{B}(p_{i,j-1}^{p+1}+p_{i,j+1}^{p})-f_{i,j}}{2(\boldsymbol{A}+\boldsymbol{B})+b}$$

其中:$\boldsymbol{A}=1/\Delta x^2$,$\boldsymbol{B}=1/\Delta y^2$,$\omega$ 为松弛因子,b 为系数,f 为已知项。

那么,如何在并行中采用超松弛迭代法呢?

3　并行求解三对角矩阵的基本原理

假设采用 m 个进程,第 1 个进程和第 m 个进程的形式分别为:

$$\boldsymbol{A}=\begin{bmatrix} a_0 & c_0 & & \cdots & & \\ b_1 & a_1 & c_1 & & & \\ & & b_{n-1} & a_{n-1} & c_{n-1} & \vdots \\ & \cdots & & b_n & a_n & c_n \end{bmatrix}\text{和}$$

$$\boldsymbol{A}=\begin{bmatrix} b_0 & a_0 & c_0 & \cdots & & \\ & b_1 & a_1 & c_1 & & \\ \vdots & & & & & \vdots \\ & & & b_{n-1} & a_{n-1} & c_{n-1} \\ & & \cdots & & b_n & a_n \end{bmatrix}$$

为了取得形式上的一致,以便计算起来通用方便,将第 1 个进程的第一个数据补充为 0,将第 m 个进程的最后一个数据也补充为 0,形式如下:

$$\boldsymbol{A}=\begin{bmatrix} 0 & a_0 & c_0 & & \cdots & & \\ & b_1 & a_1 & c_1 & & & \\ \vdots & & & & & & \vdots \\ & & & b_{n-1} & a_{n-1} & c_{n-1} & \\ & \cdots & & & b_n & a_n & c_n \end{bmatrix}\text{和}$$

$$\boldsymbol{A}=\begin{bmatrix} b_0 & a_0 & c_0 & & \cdots & & \\ & b_1 & a_1 & c_1 & & & \\ \vdots & & & & & & \vdots \\ & & & b_{n-1} & a_{n-1} & c_{n-1} & \\ & & \cdots & & b_n & a_n & 0 \end{bmatrix}$$

下面介绍具体的计算方法。

步骤 1　将矩阵分成 m 部分。任取其中一个部分,表示如下:

$$\begin{bmatrix} b_0 & a_0 & c_0 & & \cdots & & \\ & b_1 & a_1 & c_1 & & & \\ \vdots & & & & & & \vdots \\ & & & b_{n-1} & a_{n-1} & c_{n-1} & \\ & & \cdots & & b_n & a_n & c_n \end{bmatrix}\cdot\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}=\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix} \tag{3}$$

步骤 2　首先,进行向下消元的计算(上标 m 表示第 m 个进程)。

$$\boldsymbol{A}=\begin{bmatrix} b'^{(m)}_0 & a'^{(m)}_0 & c^{(m)}_0 & & \cdots & & \\ b'^{(m)}_1 & & a'^{(m)}_1 & c^{(m)}_1 & & & \\ \vdots & & & & & & \vdots \\ b'^{(m)}_{n-1} & & & & a'^{(m)}_{n-1} & c^{(m)}_{n-1} & \\ b'^{(m)}_n & & \cdots & & & a'^{(m)}_n & c^{(m)}_n \end{bmatrix}$$

其中:$b'^{(m)}_0=b^{(m)}_0$,$a'^{(m)}_0=a^{(m)}_0$,$b'^{(m)}_i=-b'^{(m)}_{i-1}\dfrac{b^{(m)}_i}{a'^{(m)}_{i-1}}$,$a'^{(m)}_i=a^{(m)}_i-c^{(m)}_{i-1}\dfrac{b^{(m)}_i}{a'^{(m)}_{i-1}}$,$(i=1,\cdots,n)$。

其次,各行都除以该行的主对角元素,使主对角线的值为 1。

$$\boldsymbol{A}=\begin{bmatrix} b''^{(m)}_0 & 1 & c'^{(m)}_0 & & \cdots & & \\ b''^{(m)}_1 & & 1 & c'^{(m)}_1 & & & \\ \vdots & & & & & & \vdots \\ b''^{(m)}_{n-1} & & & & 1 & c'^{(m)}_{n-1} & \\ b''^{(m)}_n & & \cdots & & & 1 & c'^{(m)}_n \end{bmatrix}$$

其中:$b''^{(m)}_i=\dfrac{b'^{(m)}_i}{a'^{(m)}_i}$,$c'^{(m)}_i=\dfrac{c^{(m)}_i}{a'^{(m)}_i}$。

最后,进行向上消元的计算。

$$\boldsymbol{A}=\begin{bmatrix} b'''^{(m)}_0 & 1 & & \cdots & & c''^{(m)}_0 \\ b'''^{(m)}_1 & & 1 & \cdots & & c''^{(m)}_1 \\ \vdots & & & & & \vdots \\ b'''^{(m)}_{n-1} & & \cdots & & 1 & c''^{(m)}_{n-1} \\ b''^{(m)}_n & & \cdots & & 1 & c'^{(m)}_n \end{bmatrix}$$

其中:$b'''^{(m)}_{n-1}=b''^{(m)}_{n-1}-b''^{(m)}_n\cdot c'^{(m)}_{n-1}$,$c''^{(m)}_{n-1}=-c'^{(m)}_n\cdot c'^{(m)}_{n-1}$;$b'''^{(m)}_i=b''^{(m)}_i-b'''^{(m)}_{i+1}\cdot c'^{(m)}_i$,$c''^{(m)}_i=-c''^{(m)}_{i+1}\cdot c'^{(m)}_i$,$(i=0,\cdots,n-2)$。

经过向上、向下消元之后,方程变为:

$$\begin{bmatrix} b'''^{(m)}_0 & 1 & & \cdots & & c''^{(m)}_0 \\ b'''^{(m)}_1 & & 1 & \cdots & & c''^{(m)}_1 \\ \vdots & & & & & \vdots \\ b'''^{(m)}_{n-1} & & \cdots & & 1 & c''^{(m)}_{n-1} \\ b''^{(m)}_n & & & \cdots & 1 & c'^{(m)}_n \end{bmatrix}\cdot\begin{bmatrix} x^{(m)}_0 \\ x^{(m)}_1 \\ \vdots \\ x^{(m)}_{n-1} \\ x^{(m)}_n \end{bmatrix}=\begin{bmatrix} f^{(m)}_0 \\ f^{(m)}_1 \\ \vdots \\ f^{(m)}_{n-1} \\ f^{(m)}_n \end{bmatrix} \tag{4}$$

当每个进程单向点数超过 30 个的时候,$b''^{(m)}_n\approx 0$,$c''^{(m)}_0\approx 0$(证明略)。

步骤 3　将 $c'^{(m)}_n$ 传递给下一个进程,将 $b'''^{(m)}_0$ 传递给上一个进程,任取一个组合为例,可得如下二元一次方程组:

$$\begin{bmatrix} 1 & c'^{(m)}_n \\ b'''^{(m+1)}_0 & 1 \end{bmatrix}\cdot\begin{bmatrix} x^{(m)}_n \\ x^{(m+1)}_0 \end{bmatrix}=\begin{bmatrix} f^{(m)}_n \\ f^{(m+1)}_0 \end{bmatrix} \tag{5}$$

写成方程组的形式为:

$$\begin{cases} x_0^{(m+1)}=\dfrac{f_0^{(m+1)}-f_n^{(m)}\cdot b'''^{(m+1)}_0}{1-c'^{(m)}_n\cdot b'''^{(m+1)}_0} \\ x_n^{(m)}=f_n^{(m)}-c'^{(m)}_n\cdot x_0^{(m+1)} \end{cases} \tag{6}$$

相应各个进程的 $x_0^{(m)}$、$x_n^{(m)}$都可以求解出来。

步骤 4 代回到式(4)中,因为由第 m 和第 $m-1$ 个进程,求出了 $x_0^{(m)}$,由第 m 和第 $m+1$ 个进程求出了 $x_n^{(m)}$,因此可得:

$$x_i^{(m)}=f_i^{(m)}-b'''^{(m)}_i\cdot x_0^{(m)}-c''^{(m)}_i\cdot x_n^{(m)} \quad (i=1,\cdots,n-1) \tag{7}$$

求出了各个进程每个点的值。

4 并行局部超松弛迭代的基本原理

针对上面提到的问题,本文采用了一种局部超松弛迭代的方法,即每个时间步计算的开始,将各个进程边界上的数据传递给相邻的进程(注意角点的数据),然后进行超松弛迭代,也就是说,除去边界上的点,其余的点都采用超松弛迭代法,从实际数值计算的结果来看,只是迭代的次数相对增加一些,对计算结果没有影响。值得注意的一点,为了配合方程的边界条件,对迭代的方向本文也进行了研究,结论为:从 xy 平面中心开始,向四个角点迭代,收到的效果较好。超松弛迭代示意图如图 1 所示。

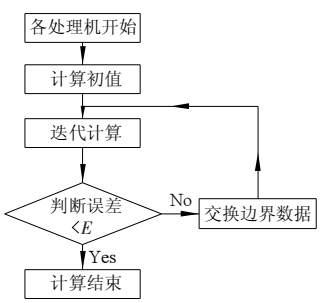

图 1 超松弛迭代示意图

5 实例验证

根据提出的方法,笔者针对不可压缩流体力学具体问题,采用 Fortran90 语言,分别编写了串行、并行程序,在曙光 TC1700 集群服务器下计算,系统为 Linux。该算例所模拟的问题为:对于不可压缩流体,在平板边界层中,入口处加入一个三维波的扰动源,该扰动源在入口处随时间而有规律的变化,之后三维波向下游传播,其幅值随空间而增长。

图 2、图 3 为同一结果的两张图,所不同的是图 2 为 y=0.76,z=1 处的串行、并行扰动空间演化情况(DNS-B 表示并行的结果,DNS-C 表示串行的结果),而图 3 为 y=0.76 处串行、并行整体扰动空间演化情况。从两张图可以看出,并行计算的结果与串行的结果完全重合,说明本文介绍的方法是正确的。

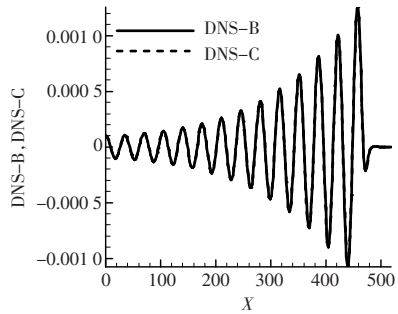

图 2 串行、并行计算结果比较图(y=0.76,z=1 处)

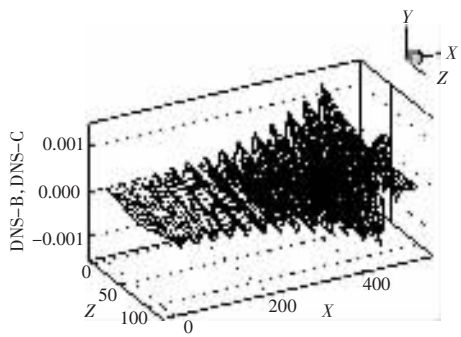

图 3 串行、并行计算结果比较图(y=0.76,三维)

本算例并行加速比为:$S=T_C/T_B$=18.26,其中:T_C 表示串行运行时间,即指在串行计算机上,程序从开始到运行结束所用的时间;T_B 表示并行运行时间,即指从并行计算开始时刻到最后的处理器完成运算所经过的时间。并行效率 E 为:E=S/p×100%=91.3%,其中:p 表示相同处理器并行计算机的个数,本文 p=20(CPU)。

6 结语

本文介绍的这种求解计算流体力学中 N-S 方程的 MPI 并行算法,解决了大型计算中计算量大、耗时长的问题,缩短了计算时间。实例证明,该方法在实际的应用中收效很好。

(收稿日期:2006 年 12 月)

参考文献:

[1] 都志辉. 高性能计算并行编程技术—MPI 并行程序设计[M].北京:清华大学出版社,2001.

[2] Karniadakis G E,Israeli M,Orszag S A.High-order splitting methods for the incompressible Navier-stokes equations[J].Journal of Computational Physics 97,1991:414-443.

[3] 傅德薰,马延文.计算流体力学[M].北京:高等教育出版社,2001.

(上接 3 页)

为了简化问题,本文将板坯排序问题转化为约束满足问题处理,利用问题的约束条件过滤变量的值域,收缩搜索空间,降低计算复杂性。应用基于域和费用值的动态变量选择和值选择启发式算法以及改进的节点互换算法确定整板顺序。仿真实验证明,约束满足+基于改进的节点互换算法具有理想的计算时间和优化效果。(收稿日期:2006 年 11 月)

参考文献:

[1] Assaf I,Chen M,Katzberg J.Steel production schedule generation[J]. International Journal of Production Research,1997,35(2):467-477.

[2] Jacobs L T,Wright R J.Optimal inter-process steel production scheduling[J].Computer Operation Research,1988,15(6):497-507.

[3] Kosiba E D,Wright J R,Cobbs A E.Discrete event sequencing as a traveling salesman problem[J].Computers in Industry,1992,19:317-327.

[4] Tang L,Liu J,Rong A,et al.A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex[J].European Journal of Operational Research,2000,124:267-282.

[5] Suh M S,Lee Y J,Kok Y K.Evaluation of ordering strategies for constraint satisfaction reactive scheduling[J].Decision Support Systems,1998,22(4):187-197.