流场类问题并行化中数组共享变量的自动搜索*

肖 骊 康继昌

(西北工业大学计算机科学与工程系 西安 710072)

摘要 本文的目的是解决流场类问题的自动并行化. 首先将流场数据均匀划分,并以 SPMD 模式对流场计算串行程序进行并行化;引入数组共享变量,着重讨论一种新方法——用递归函数实现数组共享变量的自动搜索. 用本文方法的并行化工具已初步实现.

关键词 并行处理,并行化,流场计算,数据并行,数组共享变量,递归函数.

中图法分类号 TP311

并行系统软件的落后致使应用软件的执行效率非常低,只能达到并行机峰值性能的1~25%,对 90 年代的 MPP 系统这个百分值更低,而且用户编程的难度也非常大.[1]正是这个原因,程序自动并行化成为并行处理领域中的研究热点之一. 目前,支持程序并行化的软件工具的研究已逐步展开,不断有成果推出,如 Rice 大学的 PFC 和 PED, AHPCRC 开发的 FPT,美国 APR 公司的 FORGE90[2],日本 NEC 公司的 PCASE,IBM 的 PTRAN[3],Illinois 大学 David Kuck 及同事研制的 Parafrase Ⅱ 和 SIGMACS[4],Parasoft 公司的 Express 等. 这些工具往往通过相关性分析技术挖掘并行性,难以有效实现采用时间推进的流场计算程序的并行化. 本文专门针对流场类问题的自动并行化进行研究.

1 流场计算问题模型

流场计算是包括对各种类型的流体在各种速度范围内的复杂流动、在大型计算机上进行数值模拟的一种学科. 其模型复杂多变,运算量往往很大.[5,6]但对大量网格点计算这一特点又决定了这类问题非常适合于并行计算. 流场计算是对大量网格点进行多次迭代计算,我们有如下定义:

定义 1. 帧 一帧是对某流场所有网格点的一次扫描计算.

计算每个网格点时往往需要用到周围网格点上帧计算的数据,每次迭代(每一帧)都要对所有的网格点扫描计算一遍,直到满足一定的条件,迭代结束. 而表示流场特性的几种特

---

* 本文研究得到航空科学基金资助. 作者肖骊,女,1967 年生,博士,讲师,主要研究领域为并行处理. 康继昌,1930 年生,教授,博士导师,主要研究领域为并行处理.

本文通讯联系人:肖骊,西安 710072,西北工业大学计算机科学与工程系

本文 1997-01-15 收到修改稿

性数据都是以数组形式定义的,每种特性数据对应一个数组.数组维数与流场网格维数相一致.数组下标表示网格点的位置.为描述方便,本文以二维矩形流场为例进行说明.为实现自动并行化,需做三方面工作:①自动划分流场数据和流场计算串行程序;②自动搜索划分到各结点上的程序之间需要通信的变量;③同步点的定位和通信.本文主要介绍前两项工作.

2 流场数据和流场计算串行程序的划分

根据流场计算问题模型,我们从流场数据中挖掘并行性,将原有的串行源程序以SPMD(single program multiple data)模式进行并行化,即分配到各节点上的并行程序基本与原有串行程序一致,而对流场数据进行等量划分,划分规则如下.

规则1. 二维流场有 X 和 Y 两个方向,我们选择一个方向将数据划分为 n 份(n 由用户指定,即由 n 个处理机 $P_1,P_2,\ldots,P_k,\ldots,P_n$ 并行执行).假设选择 Y 方向,有 d 个表示流场特性数据的数组(也叫网格数据)$D_1(MX,MY),D_2(MX,MY),\ldots,D_h(MX,MY),\ldots,D_d(MX,MY)$,其中 MX 和 MY 为网格在 X 和 Y 方向上的最大维数.那么 $P_k(k=1,2,\ldots,n)$ 处理机上分得的网格数据为:

$$P_k:\begin{cases}D_h(1:MX,(k-1)*U+k:k*U+k) & \text{若 } 1\leqslant k\leqslant V\\ D_h(1:MX,(k-1)*U+V+1:k*U+V) & \text{若 } V<k\leqslant n\end{cases} \tag{1}$$

其中 $U=MY/n,V=(MY/n)\text{MOD } n,h=1,2,\ldots,d$.

规则2. 将原有串行程序 F 复制 n 份($F_1,F_2,\ldots,F_k,\ldots,F_n$).要求 F 中有关 X 和 Y 方向上计算的循环控制变量一定要用 I 和 J,其它循环则不能用 I 和 J 作为循环控制变量.

规则3. 修改 $F_k(k=1,2,\ldots,n)$ 中 DO $J=J1,J2,J3$ 语句(其中 $J1$ 为上限,$J2$ 为下限,$J3$ 为步长).

对任一 J,可通过以下公式得出包含 J 的数组所应归属的 P_k,

$$k=[J/(U+1)]+1,\text{若 } k>V \text{ 则 } k=[(J-V)/U]+1 \tag{2}$$

设 $J1\in P_s,J2\in P_t$,则

$$s=[J1/(U+1)]+1,\text{若 } s>V \text{ 则 } s=[(J1-V)/U]+1$$

$$t=[J2/(U+1)]+1,\text{若 } t>V \text{ 则 } t=[(J2-V)/U]+1$$

那么:(1)对于 $k<s$ 和 $k>t$ 的程序 F_k,删去 DO $J=J1,J2,J3$ 对应的整个循环;

(2)对于 $s\leqslant k\leqslant t$ 的程序 F_k,将 DO $J=J1,J2,J3$ 改为 DO $J=J1',J2',J3$

其中

$$J1'=\begin{cases}J1 & \text{若 } k=s\\ [((k-1)*U+k-1-J1)/J3]*J3+J1+J3 & \text{若 } s<k\leqslant V\\ [((k-1)*U+V-J1)/J3]*J3+J1+J3 & \text{若 } V<k\leqslant t\end{cases}$$

$$J2'=\begin{cases}[(k*U+k-J1)/J3]*J3+J1 & \text{若 } s\leqslant k\leqslant V\\ [(k*U+V-J1)/J3]*J3+J1 & \text{若 } V<k<t\\ J2 & \text{若 } k=t\end{cases} \tag{3}$$

容易证明,根据规则1划分到各处理机上的网格数据是均衡的.

3 数组共享变量的自动搜索

流场计算时只与上帧数据有关,从而免去传统复杂的相关性分析[2,7],但分配给各处理

机的网格点数据中的边界点数据在计算时必然要用到其它处理机中网格点的上帧数据,本文把这样的数据定义为数组共享变量.

定义 2. 数组共享变量　划分到处理机 P_{k1} 上的网格数据(用数组形式定义)中有一部分数据,在计算过程中,不仅 P_{k1} 要用到,其它处理机 $P_{k2},P_{k3},\ldots$ ($k2\neq k1,k3\neq k1,\ldots$)也要用到,这部分数据是 P_{k1} 网格数据数组中的一部分,叫数组共享变量(以下简称 ASV). 对于 P_{k1} 来说,ASV 的属性定义为主属性(Master),对于 $P_{k2},P_{k3},\ldots$,ASV 的属性定义为从属性(Slave).

定义 3. 共享宽度　对任一网格点的计算一般需用周围一些网格点的数据,仍假设沿 Y 方向划分流场,且计算 $D_h(I,J)$ 需用到 $D_h(I\pm1,J\pm1),D_h(I\pm1,J\pm2),\ldots,D_h(I+W,J\pm1),\ldots,D_h(I+W,J+W)$,那么对 $P_{k1}(k1=1,2,\ldots,n)$ 来说,计算 $D_h(I,J)$ 时,当 J 为边界点时,$D_h(I\pm1,J\pm1),\ldots,D_h(I+W,J+W)$ 必是属于 P_{k2} 或/和 $P_{k3}\ldots$ ($k2\neq k1,k3\neq k1,\ldots$. 一般 $k2=k1+1$ 或 $k2=k1-1$)的数据,这里 $|W|$ 叫作共享宽度.[8]

定义 4. 共享方向　在定义 3 中,$SIGN(W)$ 即 W 的符号叫作共享方向."+"叫作正向共享,"−"叫作负向共享. 若 $k2=k1+1$,则为正向共享;若 $k2=k1-1$,则为负向共享.

我们的目标是自动搜索出 ASV,然后在每帧计算结束后将所有的 ASV 进行通信. 从而保证下帧计算所用数据的正确性.

数组共享变量的搜索规则如下:

(1) 首先建立 3 个表

a) 查找表(SL). 表中各项为:$LN,LA,RN1,RA1,RW1,RN2,RA2,RW2,\ldots$

其中 LN 为赋值语句左边变量/数组名,LA 为赋值语句左边变量/数组属性,$RN1$ 为赋值语句右边第 1 项变量/数组名,$RA1$ 为赋值语句右边第 1 项属性,$RW1$ 为赋值语句右边第 1 项带方向的宽度. $RN2,RA2,RW2$ 说明赋值语句右边第 2 项的相关信息,定义同 $RN1,RA1,RW1,\ldots$.

b) 数组共享变量总表(ASVL). 表中各项为:AN,NW,PW

其中 AN 为共享数组变量名,NW 为反向宽度,PW 为正向宽度.

c) 数组共享变量分表 ASVLk ($k=1,2,\ldots,n$). 表中各项为:$MASV,SASV$

其中 $MASV$ 为主 ASV,$SASV$ 为从 ASV.

然后,逐句扫描串行程序 F.

(2) 处理 CALL 语句时按指令执行顺序进行,其中的虚参变量全部用实参变量代替.

(3) 考虑 J 作为循环控制变量的循环体中的赋值语句,若赋值语句右边含有数组,且下标变量中含 J,则将该赋值语句信息记入 SL 中,然后调用算法 1 所示的递归函数,设赋值语句右边有 c 项,即有 $RN1,\ldots,RNc$,且是 SL 的第 r 个记录.

算法 1.

```
search()
  {for (m=1;m<=c(n);m++)
       {在 SL 的 1~r(n)-1 项中查 LN=RNm;
       if 1~r(n)-1 项中所有 LN≠RNm
         if RAm=2
           {在 ASVL 中查 AN=RNm;
            if 所有 AN≠RNm
```

```
        {给 ASVL 加入新记录,其中
          AN=RNm;
          if RWm<0 {NW=RWm,PW=0;}
          else {PW=RWm,NW=0;}}
      else/ * 找到一项 AN=RNm * /
        if 此项 NW>RWm+w(n)    NW=RWm+w(n);
        if 此项 PW<RWm+w(n)    PW=RWm+w(n)}
else/ * 1~r(n)-1 项中找到第 e 项 LN=RNm * /
{mm=m;     n=n+1;
  求得 SL 中第 e 个记录右边有 c(n)项;
  r(n)=e;     w(n)=RWm
  search();
  n=n-1;
  m=mm;}}}
```

(4)由 ASVL 和公式 1 到公式 3 生成各结点机上的 ASVLk($k=1,2,\ldots,n$).

4 结束语

我们已用本文方法初步实现了面向流场类问题的并行化工具,并通过实例验证,证明了本文方法的有效性. 今后将在此基础上更深入更广泛地研究并行化方法,更多地进行实例验证,进一步完善并行化工具. 它的实现将为我国大规模三维流场计算及类似问题的计算提供一个良好的并行计算平台.

参考文献

1 George Cybenko, David J Kuck. Supercomputers/keimenting the machine. IEEE Spectrum, 1992,(9):29~41.
2 郭克榕,唐新春. 大规模并行处理系统的程序自动并行化. 计算机世界报,1994,12,14:155~157.
3 Boleslaw K Szymanski. Parallel functional languages and compilers. New York: ACM Press, 1991. 309~391.
4 Kai Hwang. Advanced computer architecture. New York: McGraw-Hill, Inc., 1995. 456~457.
5 李晓梅,蒋增荣等. 并行算法. 长沙:湖南科学技术出版社,1992. 6~8.
6 Horst D Simon. Parallel computational fluid dynamics. Massachusetts: The MIT Press, 1992.
7 朱传琪,臧斌宇,陈彤. 程序自动并行化系统 AFT. 软件学报,1996,7(3):180~186.
8 Gyung ho Lee. Parallelizing iterative loops with conditional branching. IEEE Transactions on Parallel and Distributed Systems, 1995,6(2).

AUTOMATIC SEARCHING FOR ARRAYS' SHARED VARIABLES IN PARALLELIZING FLOWFIELD APPLICATIONS

XIAO Li KANG Jichang

(*Department of Computer Science and Engineering Northwestern Polytechnical University Xi'an 710072*)

Abstract This paper aims at parallelizing flowfield computation applications automatically. Flowfield data and serial flowfield computation programs are partitioned evenly in SPMD model. Arrays' shared variables are introduced and the recursive function that automatically searches for arrays' shared variables is discussed in detail. An automatic parallelizer has been implemented preliminarily to support the approach in this paper.

Key words Parallel processing, parallelization, flowfield computation, data parallel, arrays' shared variables, recursive function.

Class number TP311